

```
$ ls -l
total 60
drwxr-xr-x 2 user user  4096 Apr 12
drwxr-xr-x 2 user user  4096 Apr 12
-rw-r--r-- 1 user user 10000 Apr 12
-rw-r--r-- 1 user user 10000 Apr 12
-rw-r--r-- 1 user user 10000 Apr 12
-rw-r--r-- 1 user user 10000 Apr 12
-rw-r--r-- 1 user user 10000 Apr 12
-rw-r--r-- 1 user user 10000 Apr 12
-rw-r--r-- 1 user user 10000 Apr 12
-rw-r--r-- 1 user user 10000 Apr 12
```

Distributio
Open source
Gnome, KDE,
Command lin
Shell, bash
zsh, Proces
Files, Dire



Linux

The Comprehensive Guide

Michael Kofler

Rheinwerk Computing

Rheinwerk Computing

The Rheinwerk Computing series offers new and established professionals comprehensive guidance to enrich their skillsets and enhance their career prospects. Our publications are written by the leading experts in their fields. Each book is detailed and hands-on to help readers develop essential, practical skills that they can apply to their daily work.

Explore more of the Rheinwerk Computing library!



Kofler et. al.

Hacking and Security: The Comprehensive Guide to Penetration Testing and Cybersecurity

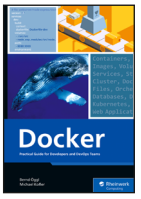
2023, 1141 pages, paperback and e-book
www.rheinwerk-computing.com/5696



Bernd Öggl, Michael Kofler

Git: Project Management for Developers and DevOps Teams

2023, 407 pages, paperback and e-book
www.rheinwerk-computing.com/5555



Bernd Öggl, Michael Kofler

Docker: Practical Guide for Developers and DevOps Teams

2023, 491 pages, paperback and e-book
www.rheinwerk-computing.com/5650



Michael Kofler

Scripting: Automation with Bash, PowerShell, and Python

2024, 470 pages, paperback and e-book
www.rheinwerk-computing.com/5851



Johannes Ernesti, Peter Kaiser

Python 3: The Comprehensive Guide

2022, 1036 pages, paperback and e-book
www.rheinwerk-computing.com/5566

www.rheinwerk-computing.com

Michael Kofler

Linux

The Comprehensive Guide

Rheinwerk Computing

Imprint

This e-book is a publication many contributed to, specifically:

Editor Meagan White
Acquisitions Editor Hareem Shafi
German Edition Editor Christoph Meister
Translation Winema Language Services, Inc.
Copyeditor Melinda Rankin
Cover Design Graham Geary
Photo Credit Shutterstock: 1929493523/© Dennis Stogsdill, 2048513402/© Andrey Suslov
Layout Design Vera Brauner
Production E-Book Kelly O'Callaghan
Typesetting E-Book SatzPro, Germany

We hope that you liked this e-book. Please share your feedback with us and read the Service Pages to find out how to contact us.

The Library of Congress has cataloged the printed edition as follows:
Names: Kofler, Michael, author.
Title: Linux : the comprehensive guide / by Michael Kofler.
Other titles: Linux (Rheinwerk Publishing). English
Description: 1st edition. | Bonn ; Boston : Rheinwerk Publishing, 2024. | Translated from German.
Identifiers: LCCN 2024009863 | ISBN 9781493224999 (paperback) | ISBN 9781493225002 (ebook)
Subjects: LCSH: Linux. | Operating systems (Computers)
Classification: LCC QA76.774.L46 K6413 2024 | DDC 005.4/46--dc23/eng/20240328
LC record available at https://lccn.loc.gov/2024009863

ISBN 978-1-4932-2499-9 (print)
ISBN 978-1-4932-2500-2 (e-book)
ISBN 978-1-4932-2501-9 (print and e-book)

© 2024 by Rheinwerk Publishing, Inc., Boston (MA)
1st edition 2024
1st German edition published 2023 by Rheinwerk Verlag, Bonn, Germany

Notes on Usage

This e-book is **protected by copyright**. By purchasing this e-book, you have agreed to accept and adhere to the copyrights. You are entitled to use this e-book for personal purposes. You may print and copy it, too, but also only for personal use. Sharing an electronic or printed copy with others, however, is not permitted, neither as a whole nor in parts. Of course, making them available on the Internet or in a company network is illegal as well.

For detailed and legally binding usage conditions, please refer to the section Legal Notes.

This e-book copy contains a **digital watermark**, a signature that indicates which person may use this copy:

Contents

Preface 31

PART I Installation

1 What Is Linux? 37

1.1 Introduction 37

1.2 Hardware Support 38

1.3 Distributions 39

1.3.1 Common Linux Distributions 41

1.4 Open-Source Licenses (GPL and Company) 44

1.4.1 Licensing Conflicts between Open- and Closed-Source Software 46

1.5 The History of Linux 46

2 Installation Basics 49

2.1 Requirements 49

2.2 BIOS and EFI 50

2.2.1 EFI System Partition 51

2.2.2 UEFI Secure Boot 51

2.3 Installation Variants 53

2.3.1 Installation Medium 53

2.3.2 Network Installation 54

2.3.3 Installation Program 54

2.3.4 Installation Location 55

2.4 Overview of the Installation Process 56

2.5 Partitioning Basics 58

2.5.1 MBR Basics 59

2.5.2 GPT Basics 59

2.5.3 Partition Names 60

2.5.4 File Systems 61

2.6 LVM and Encryption ... 61
2.6.1 Logical Volume Manager ... 61
2.6.2 Encryption ... 62
2.6.3 Limitations ... 63
2.6.4 Recommendation ... 63

2.7 Creating Linux Partitions ... 64
2.7.1 Number and Size of Linux Partitions ... 65
2.7.2 Which File System? ... 67

2.8 Setting the Scope of the Installation ... 68

2.9 Basic Configuration ... 68

2.10 System Changes, Extensions, and Updates ... 70
2.10.1 Software Installation and Package Management ... 70
2.10.2 Updates ... 70
2.10.3 Configuration ... 72

3 Installation Instructions 75

3.1 Debian ... 76
3.1.1 Installing Debian ... 78
3.1.2 Live-Image Installation ... 78
3.1.3 Standard Installation ... 80
3.1.4 Getting Started ... 83

3.2 Fedora ... 84
3.2.1 Installing Fedora ... 86
3.2.2 Getting Started ... 90

3.3 Linux Mint ... 92

3.4 Manjaro Linux ... 95

3.5 openSUSE ... 98
3.5.1 Installing openSUSE ... 100
3.5.2 Getting Started ... 103

3.6 Pop!_OS ... 104
3.6.1 Installing Pop!_OS ... 105
3.6.2 Getting Started ... 107

3.7 Ubuntu ... 108
3.7.1 Installing Ubuntu ... 110
3.7.2 Getting Started ... 114

PART II Using Linux

4 GNOME 119

4.1 Personal Assessment 120

4.2 Getting Started 121
- 4.2.1 Panel 121
- 4.2.2 Activities 122
- 4.2.3 Dock (Dash) 124
- 4.2.4 Running Programs 124
- 4.2.5 Special Features of Keyboard, Mouse, and Touchpad 126
- 4.2.6 Using the Clipboard Efficiently 128

4.3 File Manager 128
- 4.3.1 Operation 129
- 4.3.2 Removable Media 132
- 4.3.3 Access to Network Directories 133
- 4.3.4 Sharing Network Directories 134
- 4.3.5 Plugins 137
- 4.3.6 Additional Programs 137

4.4 System Configuration 138
- 4.4.1 Mouse, Touchpad, and Keyboard 139
- 4.4.2 Network Configuration 141
- 4.4.3 Online Accounts 141
- 4.4.4 Printers 141
- 4.4.5 Monitor and Projector Configuration 143
- 4.4.6 High DPI Displays 144
- 4.4.7 Colors 145
- 4.4.8 User Administration 145
- 4.4.9 Software Installation and Updates 146
- 4.4.10 Remote Maintenance 147

4.5 Fonts 148

4.6 GNOME Tweak Tool 150

4.7 GNOME Shell Extensions 152
- 4.7.1 Useful Extensions 154
- 4.7.2 Tips and Tricks 155

4.8 GNOME Shell Themes 156

4.9 Internal Details of GNOME 156
- 4.9.1 XDG Directories and Scripts 158

4.10 GNOME Classic 159

5 KDE 161

5.1 Basic Principles 161
5.1.1 Terminology 162
5.1.2 Distributions 162

5.2 Operation 163
5.2.1 Important Plasmoids 166

5.3 File Manager 167
5.3.1 Renaming Files (KRename) 169
5.3.2 External Media and Network Directories 170

5.4 KDE Configuration 170

PART III Linux Basics

6 Using the Terminal 179

6.1 Text Consoles and Terminal Windows 179
6.1.1 Terminal Window 181
6.1.2 Running Commands 184

6.2 Displaying and Editing Text Files 185
6.2.1 Text Editors 186

6.3 man and info 189

7 Bash (Shell) 193

7.1 What Is a Shell? 193
7.1.1 Other Shells 194

7.2 Configuration 195

7.3 Command Input 198
7.3.1 Expanding Command and File Names 198
7.3.2 Important Keyboard Shortcuts 200
7.3.3 Alias Abbreviations 201

7.4 Input and Output Redirection 202
7.4.1 Output Multiplication Using tee 205

7.5 Executing Commands 205

7.6 Globbing and Substitution/Expansion 208
7.6.1 Command Substitution 211
7.6.2 Single versus Double Quotation Marks 213

7.7 Variables 213
7.7.1 Environment Variables 215
7.7.2 Predefined Environment Variables 215

7.8 Bash Scripts 217
7.8.1 Example 1: grepall 217
7.8.2 Example 2: stripcomments 219
7.8.3 Example 3: applysedfile 219
7.8.4 Example 4: Backup Script 220
7.8.5 Example 5: Creating Thumbnails 221
7.8.6 Example 6: Setting Up Student Accounts 222
7.8.7 Example 7: Changing Multiple MySQL/MariaDB Databases 223

7.9 Basic Rules for Bash Scripts 223

7.10 Variables in Bash Scripts 225
7.10.1 The Scope of Variables 225
7.10.2 Variables Predefined by the Shell 226
7.10.3 Arrays 227
7.10.4 Parameter Substitution 228
7.10.5 Read Variables Using "read" 230

7.11 Branches, Loops, and Functions 230
7.11.1 If Branches 231
7.11.2 Formulating Conditions 231
7.11.3 Case Branches 233
7.11.4 For Loops 233
7.11.5 While Loops 235
7.11.6 Until Loops 236
7.11.7 Functions 236
7.11.8 Heredoc Syntax 237

7.12 Important Special Characters in Bash: Quick Reference 237

8 Zsh (Shell) 241

8.1 Installation and Configuration 242

8.2 Usage 247

8.3 Oh My Zsh 250

9 Files and Directories 253

9.1 Handling Files and Directories 253
9.1.1 Directories 254
9.1.2 Elementary Commands for Editing Files and Directories 255
9.1.3 Determining Space Requirements of Files and Directories 258
9.1.4 Wildcard Characters 259
9.1.5 Complications with Using Wildcard Characters 261
9.1.6 Hidden Files and Directories 262
9.1.7 Special Types of Files (Links, Devices, and the Like) 263

9.2 Links 264

9.3 Finding Files (find, grep, and locate Commands) 266
9.3.1 which and whereis 266
9.3.2 locate 267
9.3.3 find and grep 268

9.4 Greater Convenience with Modern Commands 271

9.5 Access Rights, Users, and Group Membership 273
9.5.1 Access Rights for Files 274
9.5.2 Access Rights for Directories 276

9.6 Special Bits and the umask Setting 279
9.6.1 Setuid, Setgid, and Sticky Bit 279
9.6.2 Owner and Group of New Files 282
9.6.3 Access Bits of New Files (umask) 283

9.7 Access Control Lists and Extended Attributes 285
9.7.1 Access Control Lists 286
9.7.2 Extended Attributes 288
9.7.3 Capabilities 289

9.8 The Linux Directory Structure 289

9.9 Device Files 293

10 Process Management 297

10.1 Starting, Managing, and Stopping Processes 297
10.1.1 Launching Programs 298
10.1.2 Foreground and Background Processes 298

10.1.3 List of All Running Processes (ps and top) 299
10.1.4 Process Hierarchy 302
10.1.5 Terminating Processes by Force (kill and xkill) 303
10.1.6 Distribution of Compute Time (nice, renice, and ionice) 304
10.1.7 Input and Output Redirection and Pipes 305

10.2 Running Processes under a Different Identity (su) 305
10.2.1 The su Command 306

10.3 Running Processes under a Different Identity (sudo) 307
10.3.1 sudo with Ubuntu 309
10.3.2 sudo with Raspberry Pi OS 310
10.3.3 sudo with Debian 311
10.3.4 sudo for RHEL and Fedora 311
10.3.5 sudo with SUSE 311

10.4 Running Processes under a Different Identity (PolicyKit) 312

10.5 System Processes (Daemons) 315
10.5.1 Kernel Threads 316
10.5.2 Starting and Stopping System Services 317

10.6 Starting Processes Automatically (Cron) 318
10.6.1 /etc/cron.hourly, .daily, .weekly, and .monthly 320
10.6.2 Anacron 322

10.7 Starting Processes Automatically (systemd Timer) 323

11 Network Tools 329

11.1 Determining the Network Status 329

11.2 Working on Other Computers (SSH) 334
11.2.1 Copying Files Securely Using scp 338
11.2.2 SSH Tunnel 339
11.2.3 SSH File System 340
11.2.4 telnet 340

11.3 Transferring Files (FTP and Others) 341
11.3.1 SFTP (Secure FTP) 342
11.3.2 wget 343
11.3.3 curl 344
11.3.4 lftp 344
11.3.5 rsync, mirror, and sitecopy 345

11.4 Lynx 346

11.5 Mutt 346

PART IV Text and Code Editors

12 Vim 351

12.1 Quick Start ... 352
12.1.1 Help ... 354
12.2 Cursor Movement ... 355
12.3 Editing Text ... 356
12.4 Search and Replace ... 360
12.5 Editing Multiple Files Simultaneously ... 361
12.6 Internal Details ... 363
12.7 Tips and Tricks ... 366

13 Emacs 369

13.1 Quick Start ... 369
13.1.1 Loading and Saving Texts and Exiting the Program ... 370
13.1.2 Online Help ... 371
13.1.3 Editing Modes ... 372
13.1.4 Keyboard ... 372
13.2 Cursor Movement ... 373
13.3 Editing Text ... 374
13.3.1 Tabs ... 376
13.3.2 Indenting and Outdenting Text Manually ... 376
13.3.3 Continuous Text ... 378
13.4 Search and Replace ... 379
13.4.1 Searching for Patterns (with Regular Expressions) ... 380
13.4.2 Search and Replace ... 381
13.5 Buffers and Windows ... 382
13.6 Special Editing Modes ... 384
13.7 Configuration ... 386
13.7.1 Setting the Font ... 386
13.7.2 Configuration at the Click of a Mouse ... 386
13.7.3 Manual Configuration Directly in .emacs ... 387
13.7.4 MELPA ... 388

PART V System Configuration and Administration

14 Basic Configuration 391

14.1 Introduction 391

14.2 Configuration of the Text Consoles 395
14.2.1 Keyboard Layout 395
14.2.2 Font 397

14.3 Date and Time 397

14.4 Synchronizing Date and Time via NTP 399
14.4.1 The systemd-timesyncd Process (Debian, Raspberry Pi OS, and Ubuntu) 400
14.4.2 Chrony (Fedora, RHEL, and SUSE) 400

14.5 Users, Groups, and Passwords 401
14.5.1 User Management 404
14.5.2 Group Management 405
14.5.3 Passwords 407
14.5.4 Interaction of the Configuration Files 411
14.5.5 Network User Management 412

14.6 PAM, NSS, and Nscd 412
14.6.1 PAM 413
14.6.2 Name Service Switch 416
14.6.3 Name Service Caching Daemon 417
14.6.4 System Security Services Daemon 418

14.7 Language Setting, Internationalization, and Unicode 418
14.7.1 Setting the Localization and Character Set 419
14.7.2 “Cannot Set/Change Locale” Error Message 423

14.8 Hardware Reference 423
14.8.1 CPU and Memory 425
14.8.2 Power Management 425
14.8.3 Interfaces and Bus Systems 426
14.8.4 Bluetooth 427
14.8.5 Hotplug System 430
14.8.6 Audio System 432

14.9 CPU Tuning 434
14.9.1 Controlling the CPU Frequency 435
14.9.2 Monitoring the CPU Temperature 437

14.10 Notebook Optimization 438
14.10.1 powertop 438

14.10.2 TLP ... 440
14.10.3 Controlling the Battery-Charging Behavior ... 442
14.10.4 Fan Control ... 443

14.11 Printing System (CUPS) ... 444
14.11.1 Sequence of the Printing Process ... 445
14.11.2 Internal Details of CUPS ... 447
14.11.3 CUPS Web Interface ... 449
14.11.4 Administrating CUPS Using Commands ... 450

14.12 Logging (Syslog) ... 452
14.12.1 rsyslogd ... 452
14.12.2 Kernel Logging ... 455
14.12.3 System Startup Log ... 456
14.12.4 Logrotate ... 456
14.12.5 Logwatch ... 457

14.13 Logging (Journal) ... 460
14.13.1 journalctl ... 461
14.13.2 Configuration ... 462

14.14 Cockpit ... 463
14.14.1 Installation ... 464
14.14.2 Security Concerns ... 465
14.14.3 Configuration ... 465
14.14.4 Operation ... 466

15 Network Configuration 469

15.1 NetworkManager ... 469
15.1.1 Configuration ... 470
15.1.2 Virtual Private Networks ... 473
15.1.3 Proxy Configuration ... 474
15.1.4 Configuring NetworkManager in Text Mode ... 475
15.1.5 Internal Details ... 476

15.2 Manual LAN and Wi-Fi Configuration ... 478
15.2.1 Activating the LAN Controller Manually ... 478
15.2.2 Retrieving DHCP Information ... 482
15.2.3 IPv6 Configuration ... 482
15.2.4 Manual Control of the Wi-Fi Controller ... 484
15.2.5 Encrypting the Wireless Network ... 485

15.3 LAN Configuration Files ... 487
15.3.1 Basic Configuration ... 487

15.3.2 DNS Configuration (resolv.conf) ... 489
15.3.3 Host Name ... 490
15.3.4 Mappings between Controllers and Network Interfaces ... 491

15.4 Distribution-Specific Configuration Files ... 491
15.4.1 RHEL and Fedora (NetworkManager) ... 492
15.4.2 Debian ... 495
15.4.3 SUSE ... 498
15.4.4 Ubuntu ... 499
15.4.5 networkd (systemd) ... 501

15.5 Zeroconf and Avahi ... 502

16 Software and Package Management 505

16.1 Introduction ... 505
16.1.1 Disadvantages of Linux Package Management ... 507
16.1.2 New Concepts ... 508

16.2 RPM Package Management ... 509
16.2.1 Basic Principles ... 510
16.2.2 The rpm Command ... 511

16.3 DNF ... 513
16.3.1 Concept ... 513
16.3.2 Configuration ... 514
16.3.3 Searching, Installing, and Updating Packages ... 516
16.3.4 AppStream ... 518
16.3.5 Additional Functions ... 520

16.4 ZYpp ... 521
16.4.1 The zypper Command ... 522

16.5 Debian Package Management (dpkg) ... 524
16.5.1 The dpkg Command ... 525

16.6 APT ... 527
16.6.1 Configuration ... 528
16.6.2 apt Command ... 530
16.6.3 The apt-get Command ... 532
16.6.4 Additional APT Commands ... 532
16.6.5 Automating Updates ... 533
16.6.6 Synaptic ... 535

16.7 Pacman ... 537

16.8 PackageKit ... 540

16.9 Firmware, BIOS, and EFI Updates ... 541

16.9.1 fwupd and fwupdmgr ... 542

16.9.2 Internal Details of Microcode Updates ... 544

16.10 Managing Parallel Installations (Alternatives) ... 545

16.11 Flatpak and Snap ... 547

16.11.1 Flatpak ... 549

16.11.2 Snap ... 550

16.11.3 AppImages ... 552

16.12 Distribution-Specific Characteristics ... 554

16.12.1 Debian ... 554

16.12.2 Fedora ... 556

16.12.3 openSUSE ... 557

16.12.4 RHEL and Clones ... 559

16.12.5 Ubuntu ... 562

17 Graphics System 567

17.1 Basic Principles ... 567

17.1.1 The X Window System ... 568

17.1.2 Wayland ... 569

17.1.3 Wayland Limitations Compared to X ... 570

17.1.4 Glossary ... 571

17.2 Graphics Drivers ... 572

17.2.1 Drivers for AMD, Intel, and NVIDIA ... 573

17.2.2 Problems of Nonfree Drivers ... 575

17.3 NVIDIA Driver Installation ... 576

17.3.1 Operation and Configuration ... 578

17.4 Determining the Status of the Graphics System ... 582

17.5 Starting the Graphics System ... 586

17.5.1 Wayland or X? ... 586

17.5.2 The Role of the Display Manager ... 587

17.5.3 Configuring the Display Manager ... 588

17.5.4 Automatic Login ... 588

17.5.5 Monitor Configuration for gdm ... 589

18 File System Administration 591

18.1 How Everything Is Connected ... 593

18.2 Formatting and Using USB Media ... 594
- 18.2.1 Formatting a USB Flash Drive or SD Card ... 595
- 18.2.2 Mounting USB Media Manually ... 596
- 18.2.3 Mounting an External Hard Disk Automatically ... 596

18.3 Device Names for Hard Disks and Other Data Media ... 598

18.4 Partitioning the Hard Disk or SSD ... 603
- 18.4.1 MBR or GPT? ... 604
- 18.4.2 Basic Rules ... 604

18.5 The parted Command ... 605
- 18.5.1 Example 1 (MBR) ... 608
- 18.5.2 Example 2 (GPT) ... 609

18.6 Partitioning Tools with a Graphical User Interface ... 610

18.7 File System Types ... 611

18.8 mount, umount, and /etc/fstab ... 616
- 18.8.1 Determining the Current State of the File System ... 616
- 18.8.2 Mounting and Unmounting File Systems Manually (mount and umount) ... 617
- 18.8.3 Mounting File Systems Automatically (/etc/fstab) ... 619
- 18.8.4 The Syntax in /etc/fstab ... 619
- 18.8.5 Bind Mounts ... 623
- 18.8.6 Automatic Mounts without /etc/fstab ... 624

18.9 Basic Principles of File Systems ... 625

18.10 The ext File System (ext2, ext3, and ext4) ... 627
- 18.10.1 Administration ... 628

18.11 The btrfs File System ... 630
- 18.11.1 Administration ... 632
- 18.11.2 Deactivating Copy-On-Write ... 633
- 18.11.3 Compressing Files ... 634
- 18.11.4 Subvolumes ... 635
- 18.11.5 Snapshots ... 636
- 18.11.6 Distributing btrfs File Systems across Multiple Devices (RAID) ... 638
- 18.11.7 Determining the Use of a btrfs File System (df) ... 640
- 18.11.8 btrfs Configuration in openSUSE ... 642
- 18.11.9 btrfs Configuration in Fedora ... 645

18.12 The xfs File System ... 646

18.13 Windows File Systems (vfat and ntfs) ... 647

18.13.1 The VFAT File System ... 649

18.13.2 The NTFS File System ... 650

18.14 Swap Partitions and Files ... 651

18.15 RAID ... 654

18.15.1 Manual Configuration Using mdadm ... 656

18.15.2 Administration ... 658

18.15.3 Replacing a Defective RAID-1 Hard Disk ... 663

18.16 Logical Volume Manager ... 665

18.17 Self-Monitoring, Analysis, and Reporting Technology ... 670

18.18 SSD TRIM ... 675

18.19 Encryption ... 676

18.19.1 Encrypt Individual Files ... 676

18.19.2 Encrypting a File System ... 677

18.19.3 Encrypting the Entire System ... 682

18.19.4 Emergency Plan ... 684

19 Grand Unified Bootloader 687

19.1 Basic Principles of GRUB ... 687

19.1.1 EFI System Startup ... 687

19.1.2 UEFI Secure Boot ... 690

19.1.3 BIOS System Boot ... 691

19.1.4 The initrd Files ... 692

19.1.5 The Future of the Boot Process ... 694

19.2 Operating GRUB (User View) ... 695

19.3 GRUB Configuration ... 696

19.3.1 Automatic Generation of grub.cfg ... 700

19.3.2 Syntax and Internal Details ... 701

19.3.3 GRUB Menu Items ... 702

19.4 Manual GRUB Installation and First Aid ... 704

19.4.1 Manual Installation and First Aid for EFI PCs ... 704

19.4.2 Changing EFI Boot Entries and Settings (efibootmgr) ... 706

19.5 systemd-boot ... 707

19.5.1 Operation ... 709

19.5.2 Configuration ... 709

20 The Init System 711

20.1 systemd 711
20.1.1 Administration 713
20.1.2 Targets 714
20.1.3 Configuration 716
20.1.4 systemd at User Level 718
20.1.5 Additional Functions 719
20.1.6 Compatibility 720
20.1.7 Documentation 721

20.2 Custom systemd Services 721
20.2.1 Custom systemd Configuration File 721
20.2.2 Example 1: Setting up Docker Containers as a Service 723
20.2.3 Example 2: Logging the Computer Startup and Shutdown 724

20.3 Shutdown, Reboot, and Halt 725

20.4 The Traditional Init-V System 726
20.4.1 Runlevel 727
20.4.2 Init-V Scripts 727
20.4.3 Links in the Runlevel Directories 728

20.5 System Startup on Fedora and RHEL 729

20.6 System Startup on Debian, Raspberry Pi OS, and Ubuntu 730
20.6.1 Raspberry Pi OS 731

20.7 System Startup on SUSE/openSUSE 732

21 Kernel and Modules 735

21.1 Kernel Modules 736
21.1.1 Commands for Module Management 737
21.1.2 Module Configuration 739
21.1.3 modprobe Syntax 741

21.2 Compiling Kernel Modules Yourself 742
21.2.1 Automating Module Updates 743

21.3 Configuring and Compiling the Kernel Yourself 745
21.3.1 Basic Principles 747
21.3.2 Installing the Kernel Code 748
21.3.3 Using Supplied Kernel Configuration Files 750

21.3.4 Configuring the Kernel Manually ... 752
21.3.5 Tools for a Manual Kernel Configuration ... 752
21.3.6 Compiling and Installing the Kernel ... 754

21.4 Kernel Live Patches ... 755

21.5 The /proc and /sys Directories ... 759

21.6 Kernel Boot Options ... 760
21.6.1 Important Kernel Boot Options ... 761
21.6.2 Symmetric Multiprocessing Options ... 763
21.6.3 Advanced Configuration and Power Interface Options ... 764

21.7 Changing Kernel Parameters ... 765

21.8 Spectre, Meltdown, and Others ... 765

PART VI Server Configuration

22 Server Installation 771

22.1 Basic Principles ... 771
22.1.1 Installation Method ... 772
22.1.2 Host Name ... 773
22.1.3 RAID/LVM Setup ... 774
22.1.4 Improving Reliability ... 776

22.2 Red Hat Enterprise Linux ... 779
22.2.1 CentOS ... 781
22.2.2 AlmaLinux, Oracle Linux, and Rocky Linux ... 782
22.2.3 Distribution Change on the Fly ... 784
22.2.4 RHEL versus Clones ... 785
22.2.5 Installation ... 786
22.2.6 Registering the RHEL Installation ... 789

22.3 Ubuntu Server ... 790
22.3.1 Installing Ubuntu Server ... 791

22.4 Debian Server Installation ... 793

22.5 Elastic Compute Cloud ... 795
22.5.1 Amazon EC2 ... 795
22.5.2 Costs ... 796
22.5.3 Getting Started ... 797
22.5.4 Setting Up the First Instance ... 798
22.5.5 SSH Access ... 801

22.5.6 EC2 Administration ... 802
22.5.7 Network Configuration ... 805
22.5.8 Amazon Linux ... 807
22.5.9 Internal Details ... 807

23 Secure Shell (SSH) 809

23.1 Installation ... 809
23.2 Configuration and Security ... 810
23.3 Fail2Ban ... 812
23.4 Authentication with Keys ... 814
23.5 Two-Factor Authentication ... 818
23.5.1 2FA with Google Authenticator ... 818
23.5.2 2FA with YubiKey ... 822
23.6 Additional Tools ... 824
23.6.1 Cluster SSH ... 824
23.6.2 Parallel SSH ... 825
23.6.3 Mosh ... 825
23.6.4 screen ... 826

24 Apache 829

24.1 Apache ... 829
24.1.1 Configuration ... 832
24.1.2 Default Character Set ... 834
24.1.3 Logrotate ... 836
24.2 Encrypted Connections (HTTPS) ... 836
24.2.1 Certificates ... 837
24.2.2 Using Self-Signed Certificates ... 838
24.2.3 Apache Configuration for HTTPS Operation ... 841
24.2.4 Snake-Oil Certificates ... 843
24.3 Let's Encrypt ... 844
24.3.1 Installing acme.sh ... 845
24.3.2 Applying acme.sh ... 845
24.3.3 SSL Settings ... 849

24.4 Setting Up and Securing Web Directories ... 851
24.4.1 Host Configuration ... 852
24.4.2 Directory Configuration ... 853
24.4.3 Securing Directories ... 855
24.4.4 Password Protection for Web Directories ... 857
24.5 Virtual Hosts ... 859
24.5.1 Setting Up Virtual Hosts ... 860
24.6 Web Access Statistics ... 861
24.6.1 GoAccess ... 862
24.7 PHP ... 865
24.8 NGINX ... 869

25 MySQL and MariaDB 873

25.1 Installation and Commissioning ... 874
25.1.1 Access Protection ... 877
25.1.2 Securing MySQL/MariaDB ... 878
25.1.3 Checking the Protection ... 880
25.1.4 Setting Up New Users ... 881
25.1.5 First Tests ... 882
25.2 Administration Tools ... 883
25.2.1 mysql ... 883
25.2.2 mysqladmin ... 884
25.2.3 MySQL Workbench ... 885
25.2.4 phpMyAdmin ... 886
25.3 Backups ... 888
25.3.1 mysqldump ... 888
25.3.2 Backup Tools and Variants ... 890
25.4 Installing WordPress ... 891

26 Postfix and Dovecot 895

26.1 Introduction and Basic Principles ... 895
26.1.1 Components of an Email Server ... 895
26.1.2 Protocols and Ports ... 897
26.1.3 The Message Flow in Detail ... 899

26.1.4 Variants and Options ... 901
26.1.5 DNS Configuration ... 902
26.1.6 Reverse DNS Entry ... 905

26.2 Postfix (MTA) ... 906
26.2.1 Installation on Debian and Ubuntu ... 907
26.2.2 Installation on EHEL ... 907
26.2.3 Configuration ... 909
26.2.4 main.cf ... 909
26.2.5 Changes to the Configuration ... 912
26.2.6 Opening Port 587 ... 913
26.2.7 Logging and Administration ... 914

26.3 Postfix Encryption (TLS/STARTTLS) ... 915
26.3.1 Sample Configuration and Keywords ... 915
26.3.2 Setting Up Custom Certificates ... 918

26.4 Postfix Accounts ... 921
26.4.1 mbox or maildir Format ... 923
26.4.2 Mail Aliases ... 924
26.4.3 Explicit Recipients List ... 925
26.4.4 Email Addresses Differing from the Linux Account ... 926
26.4.5 Virtual Domains with Shared Email Users ... 927
26.4.6 Virtual Domains with Separate Email Users ... 928
26.4.7 Virtual Domains with Virtual Mailboxes ... 929
26.4.8 Disabling the Address Verification (VRFY) ... 931

26.5 Dovecot (POP and IMAP Server) ... 931
26.5.1 Operation as POP or IMAP Server ... 937
26.5.2 SMTP Authentication for Postfix ... 938

26.6 Client Configuration ... 939

26.7 SpamAssassin ... 941
26.7.1 Automatically Moving Spam to the Junk Folder ... 944

26.8 ClamAV (Virus Protection) ... 947

26.9 SPF, DKIM, and DMARC ... 949
26.9.1 Sender Policy Framework ... 950
26.9.2 DomainKeys Identified Mail ... 951
26.9.3 OpenDKIM ... 952
26.9.4 Postfix Configuration for OpenDKIM ... 958
26.9.5 Domain-Based Message Authentication, Reporting, and Conformance ... 959

26.10 Configuration Test and Troubleshooting ... 961

27 Samba 963

27.1 Basic Principles and Terminology 963
27.1.1 Access Rights and Security Systems 965
27.1.2 Centralized or Decentralized Server Topology? 967
27.1.3 NAS Instead of Tinkering with Samba? 968

27.2 Basic Configuration and Commissioning 969
27.2.1 Configuration Changes and Status 971
27.2.2 Firewall 972
27.2.3 Securing Samba 972
27.2.4 Logging 975
27.2.5 WS-Discovery (wsdd and wsdd2) 976

27.3 Password Management 976
27.3.1 Samba Passwords 977
27.3.2 Synchronizing Samba and Linux Passwords 978
27.3.3 Mapping of Windows and Linux User Names 980
27.3.4 Putting It All Together 981
27.3.5 Working Techniques 981

27.4 Network Directories 982
27.4.1 Sharing Network Directories in GNOME and KDE 987

27.5 Example: Home and Media Server 990

27.6 Example: Company Server 993

27.7 SMB Client Access 996
27.7.1 Using Desktop Systems 997
27.7.2 Finding a Samba Server Using nmblookup 998
27.7.3 Access to Network Directories Using smbclient 999
27.7.4 CIFS-mount 1000

PART VII Security

28 Backups 1007

28.1 Déjà Dup 1007
28.1.1 Configuration and Use 1008
28.1.2 Restoring Data 1010

28.2 Back In Time 1010
28.2.1 Configuration and Use 1011
28.2.2 Restoring Data 1012

28.3 Grsync 1013
28.3.1 Configuration and Use 1014

28.4 Borg Backup 1015
28.4.1 Installation 1015
28.4.2 Usage 1015
28.4.3 Borg Backup in Scripts 1017
28.4.4 Borg Backup Using SSH 1018
28.4.5 Internal Details 1018
28.4.6 Borg User Interfaces 1018

28.5 Compressing and Archiving Files 1019
28.5.1 Compressing Files (gzip, bzip2, xz, and lzop) 1020
28.5.2 Creating Compressed Archives (tar and zip) 1021

28.6 Synchronizing Directories (rsync) 1022

28.7 Incremental Backups (rdiff-backup) 1025

28.8 Incremental Backups (rsnapshot) 1027

28.9 Backup Scripts 1030

28.10 Backups to S3 Storage 1032
28.10.1 Setting Up S3 Storage 1033
28.10.2 The aws Command 1034
28.10.3 Encryption and Example 1036

29 Firewalls 1039

29.1 Network Fundamentals and Analysis 1039

29.2 Basic Protection of Network Services 1045
29.2.1 The TCP Wrapper Library 1046
29.2.2 Starting Network Services without root Privileges 1047
29.2.3 Starting Network Services in a chroot Environment 1048

29.3 Basic Firewall Principles 1049
29.3.1 Netfilter and Nftables 1049

29.4 Firewall Configuration Tools 1050
29.4.1 Debian 1050
29.4.2 Fedora and RHEL 1051
29.4.3 firewall-cmd 1053
29.4.4 SUSE 1055
29.4.5 Ubuntu 1055

29.5 Custom Firewall Built Using nft 1057
29.5.1 Nftables: Basic Principles 1057
29.5.2 Defining Rules 1060
29.5.3 Syntax for Firewall Rules 1062
29.5.4 Simple Protection of a Web Server 1066
29.5.5 More Examples 1067

30 SELinux and AppArmor 1069

30.1 SELinux 1069
30.1.1 Internal Workings of SELinux and Usage 1071
30.2 AppArmor 1077
30.2.1 AppArmor on Debian and Ubuntu 1078
30.2.2 AppArmor on SUSE 1084

PART VIII Virtualization

31 VirtualBox 1087

31.1 Installing VirtualBox 1087
31.1.1 VirtualBox Packages of your Distribution 1088
31.1.2 VirtualBox Packages from Oracle 1089
31.1.3 Preparation Tasks 1090
31.1.4 Installing VirtualBox on Windows or macOS 1091
31.2 Setting Up VirtualBox Machines 1091
31.2.1 Setting Up a Linux Virtual Machine 1092
31.2.2 Installing Guest Additions 1095
31.2.3 Setting Up a Windows Virtual Machine 1096
31.3 Working Techniques and Configuration Tips 1096
31.3.1 Network Configuration 1097
31.3.2 SSH Access via Port Forwarding 1098
31.3.3 Data Exchange via the Clipboard 1099
31.3.4 Exchanging Data with a Shared Folder 1099
31.3.5 USB Devices in Virtual Machines 1100
31.3.6 Exporting/Importing Virtual Machines 1100
31.3.7 Grouping Virtual Machines and Running Them Invisibly 1101

31.3.8 Using VirtualBox with a High-Resolution Monitor 1101
31.3.9 Controlling VirtualBox by Command (vboxmanage) 1102
31.3.10 Enlarging Virtual Hard Disks 1102

32 QEMU and Kernel-Based Virtual Machine 1105

32.1 Basic Principles 1106
32.1.1 Internal Workings of libvirt 1109
32.1.2 Behavior when Rebooting the Host System 1111
32.1.3 Virtual Hardware 1111

32.2 Virtual Machine Manager 1113
32.2.1 Setting Up a New Virtual Machine 1116
32.2.2 Stopping Virtual Machines 1118
32.2.3 Windows Installation 1119

32.3 libvirt Commands 1119
32.3.1 virsh 1119
32.3.2 virt-clone 1123
32.3.3 virt-sysprep 1123
32.3.4 virt-viewer 1124
32.3.5 virt-top 1125

32.4 Integrating Virtual Machines into the LAN (Network Bridge) 1125
32.4.1 Configuring the Network Bridge on the Host Computer 1126
32.4.2 IP Forwarding 1128
32.4.3 Network Configuration on a Root Server 1128
32.4.4 Configuring the Virtual Machine 1129
32.4.5 MAC Trouble 1129

32.5 Direct Access to the Contents of an Image File 1130
32.5.1 Access to Partitioned RAW Images in the Host System 1130
32.5.2 libguestfs Tools 1131
32.5.3 Converting an Image Format 1134
32.5.4 Enlarging an Image 1134
32.5.5 Reducing an Image File 1135

33 Windows Subsystem for Linux 1137

33.1 Checking Out WSL 1138
33.1.1 Using WSL 1139
33.1.2 File System 1141
33.1.3 Running Programs in Graphics Mode (WSLg) 1143

33.2 WSL Network Integration 1145

33.3 The wsl Command and WSL Configuration 1146
33.3.1 Global WSL Configuration 1147
33.3.2 Linux-Specific Configuration 1148
33.3.3 Enlarging the WSL2 Disk 1149

The Author 1151
Index 1153
Service Pages | Legal Notes I

Preface

Linux began as a hobby project of the Finnish programmer Linus Torvalds, and today it dominates many segments of the IT market. Surprisingly, only IT professionals are aware of the triumph of Linux: billions of people use Android smartphones, cloud infrastructure servers, Wi-Fi routers, and Internet of Things (IoT) devices without knowing that almost all of these devices run Linux.

Linux is ubiquitous...

The reason Linux leads a shadowy existence is its failure in the desktop PC segment: the masses are annoyed by Windows on a daily basis, while a minority have devices that run macOS. But what about Linux? Linux runs (almost) exclusively on the notebook computers of technology students, scientists, or administrators.

... just not on the desktop PC

There are several reasons for this: Linux does not offer *one* desktop system, but many. This fragmentation has meant that no system really works perfectly. GNOME is most likely to succeed in satisfying the needs of "common" users.

But there are also problems on the hardware side. Even though Linux runs smoothly on many devices, new notebook computers in particular often cause trouble: sometimes Bluetooth doesn't work properly, sometimes the installation of the graphics driver causes problems, and so on.

Now of course I don't want to scare you off here in the preface! On the contrary, in this book I will show you how efficiently you can use Linux. Linux offers a versatility and flexibility that other operating systems struggle to match. It's also an extremely secure system.

What makes Linux so successful in all segments except the desktop PC? The free and open availability of the source code of Linux and most of the programs running on Linux makes it possible to adapt Linux to new challenges faster and more straightforwardly than other operating systems—no matter whether for IoT, artificial intelligence, software for self-driving cars, or simulation models for climate research.

The key to its success: open source

What This Book Can Do—and What It Can't

This book introduces you to Linux from the ground up. The range of topics extends from the installation of Linux on a notebook computer through desktop use to server and cloud deployment.

It is especially important to me that you learn not only to use Linux, but also to understand it: detailed basic chapters explain how to use Linux in the terminal, how to optimally configure Linux, and why Linux works the way it does. After reading these chapters, you will know not only Linux itself, but also the philosophy of Unix/Linux—in a sense, *the Linux way to do it*.

Despite its nearly 1,200 pages, this book cannot describe every problem that may arise when running Linux. Especially with hardware incompatibilities, I usually can't help, simply because I don't have the possibility to try out various Linux distributions on every conceivable hardware setup. To enable you to research individual questions yourself, the book contains many links to further sources.

Have fun! The more you learn about the Linux world, the more Linux becomes *your* operating system. I hope you enjoy experimenting with and using Linux!

Michael Kofler

How This Book Is Organized

This book is divided into eight parts:

- Part I explains what Linux actually is and provides the basic knowledge you need for an optimal and secure *installation*. Here you can find concrete installation instructions for quite a few distributions: Debian, Fedora, Linux Mint, Manjaro, openSUSE, Pop!_OS, and Ubuntu.
- Part II covers Linux on the *desktop PC*. Here you will learn about different desktop systems. My focus is clearly on the beginner-friendly GNOME.
- In Part III, you will get to know the *terminal*. Over several chapters, you will learn which commands to use to search the file system, and how to use the `bash` command interpreter.
- Part IV focuses on various *text editors*, specifically Vi and Emacs.
- Part V is dedicated to *system configuration*. Even if there are problems with your hardware or if you have very special requirements, you will learn here how to administrate the file system, configure the graphics system, install and update software packages, configure the system startup, and set up or recompile the kernel and its modules.
- Part VI shows how you can use *Linux as a server*. A new chapter describes the installation of Red Hat Enterprise Linux (RHEL) and its clones, as well as Debian and Ubuntu Server. I will describe sensible RAID/Logical Volume Manager (LVM) setups as well as the optimal cloud configuration. The other chapters explain the configuration of important server pro-

grams, including Secure Shell (SSH), Apache, MySQL/MariaDB, Postfix and Dovecot, and Samba.

- Part VII deals with various aspects of *security*. Here you will learn how to perform backups and protect your servers using firewalls, SELinux or AppArmor.
- Part VIII is about different types of *virtualization*: Here you will get to know the VirtualBox desktop virtualization system as well as the server virtualization system, KVM. Finally, you'll learn that you can now run Linux directly on Windows using the Windows Subsystem for Linux (WSL).

Formal Aspects

Commands

In this book, the parts of a command that actually have to be entered are highlighted in bold.

So, in the following example, you only need to type "ls *.tex" to display the list of all `*.tex` files in the current directory:

```
user$ ls *.tex
article.tex
...
```

Multiline commands

If individual commands do not fit in one line, they are distributed across two or more lines by means of the \ character, a character permitted on Linux to perform multiline command input. Of course, you can also type the command on a single line without using \.

root

Some commands can be executed only by the `root` system admin. In this case, the command prompt is displayed as `root#`:

```
root# systemctl restart apache2
```

sudo

The easiest way to run commands with `root` privileges on many distributions is to use `sudo`. On Ubuntu, this is in fact the only possible way:

```
user$ sudo systemctl restart apache2
Password: ********
```

MiB versus MB

In computer science, it's common to calculate with powers of two. One megabyte is therefore not one million bytes, but 2^{20} bytes—that is, exactly 1,048,576 bytes. To emphasize this fact, the International Electrotechnical Commission (IEC) recommends the use of the units KiB, MiB, GiB, and TiB (see *https://en.wikipedia.org/wiki/Byte*).

Distributions The countless Linux distributions can be grouped into families according to their origin. Much of the information in this book therefore applies to multiple distributions:

- Arch Linux: Also applies to Manjaro
- Debian: Mostly also valid for Raspberry Pi OS, partly also for Ubuntu
- RHEL: Also applies to all clones (such as AlmaLinux, Oracle Linux, and Rocky Linux)
- Ubuntu: Largely also applies to Kubuntu, Linux Mint, Pop!_OS, and so on

PART I

Installation

Chapter 1
What Is Linux?

To answer the question in this chapter's title, I will first explain some important terms that are used repeatedly throughout the book, such as *operating system*, *Unix*, *distribution*, *kernel*, and so on. A brief overview of the features of Linux and the available programs then will help clarify how wide the range of options for using Linux is.

I want to start with a brief excursion into the history of Linux. Of central importance here is, of course, the *GNU General Public License* (GPL for short), which specifies the conditions under which Linux may be distributed. Only the GPL makes Linux a free system, whereby *free* means more than simply *free of charge*.

1.1 Introduction

Linux is a Unix-like operating system. The most important difference compared to historical Unix systems is that Linux may be freely copied including the complete source code.

Operating system

An operating system is a bundle of programs that realize the basic functions of a computer. This includes managing the keyboard, screen, and mouse as well as the system resources (CPU time, memory, SSDs, etc.). You need an operating system to be able to start an application program and to save your own data in a file. Popular operating systems include Windows, Linux, macOS, Android, and iOS.

Unix versus Linux

Long before Windows, Linux, and smartphones, there was Unix. From a technical point of view, this operating system was ahead of its time: real multitasking, separation of processes, access rights for files, and so on. However, Unix initially offered only a very minimalist user interface, had high hardware requirements, and ran on expensive workstations only.

Today, Linux has replaced Unix. Large parts of the internet are operated by Linux servers. Linux runs as Android on smartphones, in routers and on network-attached storage (NAS) hard drives, and in supercomputers: in fact, the 500 fastest computers in the world today *all* run Linux (*https://top500.org/statistics/list*).

Kernel

Strictly speaking, the term *Linux* only refers to the *kernel*, the innermost part (i.e., the core) of an operating system with very basic functions, such as memory management, process management, and control of the hardware. The information in this book refers to Linux kernel 6.*n*.

1.2 Hardware Support

Linux supports almost all common PC hardware and also runs on other hardware platforms, such as smartphones or embedded devices. Nevertheless you need to pay attention when buying a new computer. There are some hardware components that often cause problems when interacting with Linux:

- **Graphics cards**
  Almost all graphics cards on the market or graphics cores integrated into the CPU work on Linux. For many Linux users without special demands on the graphics system, Intel or AMD CPUs with built-in graphics cores are the ideal solution. Graphics cards from NVIDIA, on the other hand, often require additional drivers so that the card can be used perfectly. The installation of these drivers can cause problems.
- **Wi-Fi and network adapters**
  Wi-Fi and LAN controllers rarely cause problems. Only very new models are sometimes not yet supported by Linux.
- **Energy-saving functions**
  New notebooks in particular often have significantly shorter battery runtimes when they run on Linux rather than on Windows. This issue results from the fact that the interaction of various energy-saving functions requires optimal drivers, which are often not available for Linux at all or only one or two years after market launch.

So *prior to* buying a new computer or hardware extension, you should make sure that all components are supported by Linux. Also, an internet search for "linux <hardwarename>" can be useful.

A Little Older Is Better

As a rule, I give brand-new notebooks a wide berth when buying them, no matter how tempting the specifications may be. They all too often cause driver problems and create extra work during installation and configuration. You can save money and avoid getting angry during the configuration process if you opt for a previous year's model!

1.3 Distributions

We still haven't provided a comprehensive answer to the introductory question: What is Linux? Many users are not interested in the kernel. For them, the term *Linux*, as it is colloquially used, includes not only the kernel but also a huge bundle of included programs: countless commands, a desktop system (e.g., KDE or GNOME), LibreOffice, Firefox, GIMP, and numerous programming languages and server programs (web server, mail server, etc.).

A Linux distribution is the unit that consists of the actual operating system (kernel) and many additional programs. A distribution allows for a quick and convenient installation of Linux. Most distributions can be downloaded free of charge from the internet.

Distributions mainly differ from each other in the following aspects:

- **Scope and up-to-dateness**
  The number, selection, and up-to-dateness of the included programs and libraries vary greatly. Some distributions, such as Debian, for example, are deliberately based on somewhat older, stable versions.
- **Installation and configuration tools**
  The provided programs for installation, configuration, and maintenance of the system help to set the configuration files. This can save a lot of time.
- **Configuration of the desktop (KDE and GNOME)**
  Some distributions enable the user to choose from among KDE, GNOME, and other desktop systems. The detailed configuration and visual design also vary depending on the distribution.
- **Hardware support**
  Linux gets along with most PC hardware components. Nevertheless, there are differences in detail among the distributions, especially when it comes to integrating non-open-source drivers (such as for NVIDIA graphics cards) into the system.
- **Updates**
  You can only run a Linux distribution safely as long as you can get updates. After that, you should switch to a new version of the distribution. For this reason, it is important to know for how long updates will be available for a distribution. In this context, the basic rule usually is: the more expensive the commercial support, the longer the update period. Let's look at some examples (as of summer 2023):
  - Debian: Three years (five years with limitations)
  - Fedora: 13 months

  - openSUSE: Approximately 18 to 24 months
  - Red Hat Enterprise Linux (RHEL): 10 years (up to 13 years with limitations)
  - RHEL clones: Up to 10 years
  - SUSE Enterprise Server: 10 years (up to 13 years with limitations)
  - Ubuntu LTS: Three to five years (with Pro upgrade, 10 years)
  - Ubuntu (other versions): Nine months

- **Rolling release**
  All distributions listed previously explicitly distinguish between versions. Thus, Ubuntu 23.10 contains different versions of GNOME, LibreOffice, and GIMP than Ubuntu 24.04, for example.

  However, there are also distributions that use the *rolling release* model, such as Arch Linux or openSUSE Tumbleweed. This model makes sure that you will always get the latest version of *each* installed software component with updates. This sounds pretty useful, but it can cause stability problems, which is why rolling release distributions are not common in the server area. They appeal more to advanced Linux users who develop software or administrate systems and who have no problem adjusting the odd configuration file after an update if something no longer works.

- **Live system**
  Many distributions allow for operating Linux directly from a USB stick. This enables you to easily perform a trial and error run. In addition, such live systems offer a good opportunity to repair a defective Linux system or to reinstall the respective distribution.

- **Target platform (CPU architecture)**
  Many distributions are only available for Intel- and AMD-compatible processors. But there are also distributions for other processor platforms (ARM, SPARC, etc.).

- **Support**
  If you purchase a commercial distribution, you will receive help with installation and operation.

- **License**
  Most distributions are available free of charge. However, some distributions have restrictions in this respect. For example, Red Hat and SUSE enterprise distributions only allow registered customers to access the update system. You pay not for the software itself, but for the services offered.

Linux Standard Base (LSB)

The Linux Standard Base (LSB) project defines rules to create a common denominator between distributions. Most distributions are LSB compliant (see *https://wiki.linuxfoundation.org/lsb/start*).

1.3.1 Common Linux Distributions

The following overview of the most important distributions available is designed to give you some initial guidance. I present them in alphabetical order. I do not claim this to be a complete listing.

AlmaLinux

AlmaLinux is a RHEL clone—that is, a distribution compatible with RHEL. Together with Rocky Linux, AlmaLinux has succeeded CentOS Linux.

Android

Android is a platform developed by Google for mobile devices and tablets. Android has thus helped Linux achieve the world dominance that Linux developers have joked about in the past. However, Android is not suitable for a PC installation and is therefore not a "real" distribution.

Arch Linux

Arch Linux is a rolling release distribution optimized for technical users. Due to the relatively complex installation, which has to be carried out in text mode, beginners usually avoid Arch Linux. Nevertheless, the wiki at *https://wiki.archlinux.org* is one of the best sources for Linux configuration details on the web.

Arch Linux derivatives such as *Manjaro* and *EndeavourOS* with graphical installation and configuration programs have recently pushed Arch Linux into the top 10 list at *https://distrowatch.com*.

CentOS

CentOS was a free variant of RHEL and had a huge install base. However, Red Hat announced the end of CentOS in its current version in December 2020.

CentOS Stream

CentOS Stream is supposed to be the successor to CentOS. However, this variant differs from the original CentOS in two important details. First, the maintenance period is much shorter: only four to five years instead of the previous 10 years.

Second, most package updates (except security updates, which are subject to a nondisclosure agreement) are first released for CentOS before being deployed for RHEL. At first glance, this seems to be an advantage. As a matter of fact, however, there is no longer full compatibility with RHEL. This detail also turns CentOS users into beta testers for updates. CentOS Stream is unsuitable for long-term production use.

Chrome OS

Chrome OS is developed by Google, just like Android. It is optimized for notebooks and its use requires an active internet connection. The user interface is based on the Google Chrome web browser. Chrome OS does not currently play a big role in Europe, but it does in the education market in

the US, where inexpensive Chromebooks (notebooks running the Chrome OS) are often used in schools.

Debian *Debian* is the oldest entirely free distribution. It is compiled by dedicated Linux developers, and its adherence to the rules of the game of "free" software is a high priority. The strict interpretation of this philosophy has led to delays several times in the past.

Debian is intended for advanced Linux users and has a large market share in server installations. Compared to other distributions, Debian is strongly optimized for maximum stability and therefore often does not contain the latest program versions. However, Debian is available for nine hardware platforms. Many other distributions are derived from Debian, such as Raspberry Pi OS and Ubuntu.

Fedora *Fedora* is the free development branch of Red Hat Linux, and it's supported and managed by Red Hat. For Red Hat, Fedora is a kind of playground where new features can be tried out without putting the stability of the enterprise versions at risk. Programs that work well on Fedora will later be integrated into the enterprise versions. Among technically interested Linux fans, Fedora is popular because this distribution often plays a pioneering role: new Linux features are often found in Fedora first and only later in other distributions. New Fedora versions are released once every six months. Updates are discontinued one month after the release of the version after the next one; the 13-month lifespan is very short.

Kali Linux Based on Debian, *Kali Linux* includes a huge collection of hacking and pentesting tools. This distribution is considered *the* toolbox for hackers and security experts.

openSUSE *openSUSE* is a free Linux distribution based on the enterprise versions of SUSE, but specifically intended for home users and developers. Note that in this book I often simply refer to *SUSE* when I actually mean both the commercial offerings of SUSE and the community offering, openSUSE. In many details, the distributions behave in the same way.

Oracle Oracle provides a variant of RHEL called *Oracle Linux*. This is a permissible approach due to the open-source licenses. Technically speaking, there are only a few differences from RHEL, but the Oracle variant is cheaper and even available for free without support.

Raspberry Pi OS *Raspberry Pi OS* is the standard distribution for the popular Raspberry Pi minicomputer. Raspberry Pi OS is based on Debian but has been specifically adapted and extended for the Raspberry Pi.

Red Hat Acquired by IBM in 2018, *Red Hat* is the best-known and most successful Linux company in the world. Red Hat distributions dominate the American

market in particular. The package management function, which is based on the Red Hat package format (RPM), has been adopted by many other distributions.

Red Hat is predominantly focused on enterprise customers. The enterprise (RHEL) versions are comparatively expensive. They are characterized by high stability and a ten-year update period. For Linux enthusiasts and developers looking for a Red Hat–like system at zero cost, AlmaLinux, CentOS Stream, Fedora, Oracle Linux, and Rocky Linux are available.

Rocky Linux

After the end of CentOS Linux in its previous design, new RHEL clones have been fighting to succeed it. Rocky Linux is one of the most important contenders, along with AlmaLinux, and has recently gained a lot of popularity as well as support from well-known companies.

SUSE

After various acquisitions, SUSE has become the most important German Linux company. Its main product, *SUSE Enterprise*, is primarily anchored in the European market. Since 2021, SUSE has been publicly listed. In August 2023, however, financial investor EQT announced its intention to delist the company again.

Ubuntu

Ubuntu is currently the most popular distribution for home users. Ubuntu uses Debian as a base, but it's better optimized for desktop users (its motto: *Linux for human beings*). The free distribution is published every six months. For ordinary versions, updates are provided over nine months. For the biennial versions with long-term support (LTS), updates are available for three to five years.

For commercial customers, Canonical offers various support packages, including *Ubuntu Pro*. This paid upgrade extends the update period of LTS versions to ten years. Canonical has thus become one of the world's most important Linux companies, especially in the server and cloud sector.

Ubuntu derivatives

There are a lot of official and unofficial variants of Ubuntu. The most well-established and widely used ones are *Ubuntu Server*, *Kubuntu*, *Xubuntu*, *Ubuntu MATE*, and *Linux Mint*. The US company System76 maintains *Pop!_OS*, an Ubuntu variant that is especially optimized for notebooks and supports NVIDIA graphics cards particularly well. *KDE Neon* is another interesting option: this distribution combines Ubuntu LTS with always-up-to-date KDE packages and is therefore popular with KDE fans.

Other distributions

Besides the "big" distributions listed here, there are numerous compilations of miniature systems available on the internet. These are primarily designed for special tasks, such as maintenance work (emergency systems) or to be able to use a Linux system without actually installing it (live systems). Popular representatives of this Linux genre include *Parted Magic*, *Slax*, and *TinyCore*.

A pretty good overview of all currently available Linux distributions, commercial or otherwise, can be found at *https://distrowatch.com*.

The agony of choice

Making a recommendation for a specific distribution is difficult. For Linux beginners, it is usually advantageous to choose a widely used distribution such as Debian, Fedora, openSUSE, or Ubuntu. Another good choice is Linux Mint. A lot of information about these distributions is available on the internet as well as in book and magazine stores. When problems arise, it is comparatively easy to find help.

Commercial Linux users or server admins need to decide if they are willing to spend money for professional support. In this case, there is little to be said against the market leaders: Red Hat, Ubuntu, and SUSE. Otherwise, AlmaLinux, Debian, Oracle Linux, Rocky Linux, and Ubuntu are good free alternatives.

1.4 Open-Source Licenses (GPL and Company)

The basic idea of the open-source model is that the source code of programs is freely available and may be extended or changed by anyone. However, there is also an obligation associated with it: anyone who uses open-source code to develop their own products must also freely redistribute all code again.

Incidentally, the open-source idea in no way prohibits the sale of open-source products. At first glance, this seems to be a contradiction. But in fact, the freedom of open source refers more to the code than to the finished product. In addition, the free availability of the code also regulates the pricing of open-source products: only those who offer additional services (manuals, support, etc.) alongside the compilation of an open-source program will survive. As soon as the price is not in reasonable proportion to the services, other companies will be found that do it cheaper.

General Public License (GPL)

The goal of open-source developers is to create software for which the source code is freely available and will remain so. To prevent misuse, many open-source programs are protected by the GPL. The GPL is backed by the Free Software Foundation (FSF), an organization founded by Richard Stallman to make high-quality software freely available. Incidentally, Richard Stallman is also the author of the Emacs editor, which is described in Chapter 13.

The core statement of the GPL is that while anyone may modify the code and even sell the resulting programs, at the same time the user/buyer has the right to the complete code and may also modify it and redistribute it

free of charge. Every GNU program must be distributed together with the full GPL text. The GPL thus precludes anyone from further developing and selling a GPL program *without* making the changes publicly available. Every further development is therefore a benefit for *all* users. The full text of the GPL can be found at *https://gnu.org/licenses/gpl.html*.

The concept of the GPL is quite simple to understand, but in detail questions arise again and again. Many of them are answered at *https://gnu.org/licenses/gpl-faq.html*. If you think you understand all of it, you should check out the GPL quiz at *https://gnu.org/cgi-bin/license-quiz.cgi*.

Lesser General Public License (LGPL)

In addition to the GPL, there is also a Lesser GPL (LGPL) variant available. The essential difference from the GPL is that a protected library may also be used in commercial products whose code is *not* freely available. Without the LGPL, GPL libraries could only be used again for GPL programs, which in many cases would be an undesirable restriction for commercial programmers.

Other licenses

By all means, not all parts of a Linux distribution are subject to the same copyright terms! Although the kernel and many tools are subject to the GPL, other legal conditions apply to some components and programs:

- **MIT and BSD licenses**
  The MIT and BSD licenses allow for commercial use of the code *without* an obligation to publicly distribute changes. The licenses are thus much more liberal than the GPL and more comparable to the LGPL.
- **Dual licenses**
  Dual licenses apply to some programs. For example, you can use the MySQL database server free of charge for open-source projects, on your own web server, or for in-house use under the GPL. If, on the other hand, you want to develop a commercial product based on MySQL and sell it together with MySQL without making your source code available, then the commercial license will come into play. In this case, the distribution of MySQL is subject to a fee.
- **Commercial licenses**
  Some programs are subject to a commercial license but may still be used free of charge. The source code then remains a company secret. Two examples are Microsoft Teams and Skype.

Some distributions mark products for which use or redistribution might cause licensing problems. In Debian, such programs are included in the *nonfree* package source.

The thicket of countless, more or less “free” licenses is difficult to navigate. There is a wide area between the sometimes fundamentalist interpretation

of *free* in the sense of the GPL and the codified provisions of some companies that want to call their software product free (because this is fashionable at the moment), but in reality want to retain unrestricted control over the code. A good introduction to the topic can be found at *https://opensource.org*.

1.4.1 Licensing Conflicts between Open- and Closed-Source Software

Open-source licenses for developers

If you develop programs and want to sell them together with Linux or in combination with open-source programs or libraries, you need to delve deeper into the sometimes confusing issue of different software licenses. Many open-source licenses only permit redistribution if you also make your source code freely available under an open-source license. The more open-source components with different licenses your program is based on, the more complicated the distribution becomes.

However, there are exceptions that facilitate the commercial use of open-source components. For example, consider Apache and PHP: you may freely distribute these programs in combination with a closed-source program.

GPL problems with hardware drivers

Some proprietary drivers for hardware components (e.g., for NVIDIA graphics cards) consist of a small kernel module (open source) and various external programs or libraries whose source code is not available (closed source). The only purpose of the kernel module is to establish a connection between the kernel and the closed-source driver.

These drivers are a good thing from the point of view of many Linux users: they are available free of charge and make it possible to use various hardware components for which there are either no drivers at all or at least no complete open-source drivers for Linux.

The question, however, is whether or to what extent the closed-source drivers violate this license due to their close integration with the kernel, which is subject to the GPL. Many open-source developers are reluctant to tolerate the drivers. A direct distribution with GPL products is not permitted, which is why users must usually download and install the drivers themselves.

1.5 The History of Linux

1982: GNU

Because Linux is a Unix-like operating system, I should actually start with the history of Unix at this point, but there is no space for that here. Instead, this history lesson begins with the founding of the GNU Project by Richard Stallman. GNU stands for *GNU is not Unix*. In this project, open-source tools

have been developed since 1982. These include the GNU C compiler, the Emacs text editor, and various GNU utilities such as `find` and `grep`, among others.

1989: GPL

It was not until seven years after the start of the GNU Project that the time was ripe for the first version of the General Public License. This license ensures that free code remains free.

1991: Linux kernel 0.01

The very first parts of the Linux kernel (version 0.01) were developed by Linus Torvalds. He released his code on the internet in September 1991. Programmers around the world quickly became interested in the idea and wrote extensions. When the kernel of Linux allowed the execution of the GNU C compiler, the full range of GNU tools was also available. Other components included the Minix file system, network software from BSD-Unix, MIT's X Window System and its XFree86 port, and others.

Thus, Linux is not the work of only Linus Torvalds. Instead, Linux is backed by a lot of dedicated people who produce free software in their leisure time, as part of their studies, or in return for payment from companies such as Google, IBM, or HP.

1994: First distributions

Computer science geeks at universities were able to download, compile, and install Linux and its components themselves. However, Linux only became widely used with Linux distributions, which packaged Linux and the software developed around it on floppy disks or CD-ROMs and provided them with an installation program. Four of the distributions created at that time still exist today: Debian, Red Hat, Slackware, and SUSE.

1996: Penguin

In 1996, Tux the penguin became the Linux logo.

1998: Microsoft sees Linux

With the rapid triumph of the internet, the spread of Linux also increased, especially on servers. Steve Ballmer's legendary statement became a kind of accolade for Linux: *Microsoft is worried about free software.* A year later, Red Hat went spectacularly public.

The common denominator of Amazon (founded in 1994), Google (1998), Wikipedia (2001), Facebook (2006), and Dropbox (2007) is that all of these companies or organizations rely on Linux for their server operations. The internet, as it has been developing since the 2000s, and the cloud infrastructure that has grown out of it are inconceivable without Linux.

2009: Android

With the Android platform, Google first brought Linux to cell phones (2009), then to tablets and TVs.

2012: Raspberry Pi

In 2012, the Raspberry Pi minicomputer captured the hearts of electronics hobbyists. For only around $40, you can conduct your own hardware experiments with the Raspberry Pi, enter the world of home automation,

and operate a media center or a home server. The Raspberry Pi makes embedded Linux a mass phenomenon.

2018: IBM aquires Red Hat, Microsoft, and Linux

In 2018, IBM acquired Red Hat for a handsome $34 billion. At the same time, under the leadership of Satya Nadella, Microsoft turned more and more to the open-source idea and Linux. The *Windows Subsystem for Linux* allows Linux to run directly on Windows. With its purchase of GitHub, which also happened in 2018, Microsoft now dominates the most important repository for open-source projects.

2021: SUSE goes public

In May 2021, SUSE became a publicly traded company. The company's initial valuation of just under €6 billion is quite substantial. However, long-term success is not forthcoming, and thus a withdrawal from the stock exchange was announced in summer 2023.

Chapter 2
Installation Basics

This chapter provides an overview of how to install a Linux system on a notebook computer or a PC with an Intel-compatible processor. The chapter does not refer to a specific distribution, but describes essential installation steps (e.g., partitioning the hard disk) in a general manner and provides the necessary basic knowledge. Specific details on the installation of some selected distributions will follow in the next chapter.

The installation has become easier and easier in recent years. In an ideal scenario—that is, if you use standard hardware and there is enough storage space available for Linux—30 minutes should be enough to set up a working Linux system. The installation becomes more difficult if space is to be created for a parallel Linux distribution in an already existing Windows installation. However, problems can also arise when unusual or completely new hardware must be supported.

2.1 Requirements

To install Linux, several requirements must be met:

- You need a PC or notebook computer with an Intel-compatible processor. This includes all common 64-bit processors from Intel or AMD. There are also Linux distributions available for systems with other processor architectures (such as ARM), but these are not the subject of this chapter.
- You need enough space on your SSD or hard disk to set up Linux partitions. The question as to how much is "enough" depends on the distribution and, of course, on how many programs you install and what personal data you want to store (photos, videos, etc.), if you want to use the computer to develop software, and so on. My recommendation is that you need at least 30 GiB just to try out Linux.
- You need hardware components that are recognized and supported by Linux. Currently, this is the case for much of the standard hardware. New wireless network adapters or graphics cards are most likely to cause problems (see Chapter 1, Section 1.2).

Distributions for Legacy PCs and Virtual Machine Installations

As I mentioned in the previous chapter, there are minimal distributions that have much lower hardware requirements. However, in this chapter I assume that you are installing a common distribution like Debian, Fedora, Linux Mint, openSUSE, or Ubuntu. Their hardware requirements are similar to those for Microsoft Windows.

If you use virtualization programs such as VirtualBox, VMware, or UTM, then you can install and run Linux inside Windows or macOS in a virtual environment. This simplifies the installation process considerably, but also reduces the functionality (limited hardware access, slower graphics display, etc.).

2.2 BIOS and EFI

BIOS

For decades, the BIOS (basic input/output system) was responsible for initializing PCs and notebook computers. This is a program that is executed immediately after the computer has been switched on. BIOS is responsible for detecting the hardware components, configuring the hardware, and starting the operating system. The term BIOS has been in use for these functions for almost 50 years!

EFI and UEFI

The traditional BIOS carries a lot of legacy issues with it. That's why Intel started developing the EFI (Extensible Firmware Interface) successor to BIOS in 1998. Later, AMD, Apple, Microsoft, and others participated in further development, after which the software got the new abbreviation UEFI (Unified Extensible Firmware Interface). The abbreviations EFI and UEFI have been used synonymously ever since, including in this book: when talking about EFI on modern motherboards, we always mean UEFI.

The biggest advantage compared to BIOS is that EFI allows you to choose which of several installed operating systems you want to start when you boot the computer. This has greatly simplified the parallel operation of Windows and Linux. In the BIOS past, the Grand Unified Bootloader (GRUB) program had to make this selection.

EFI has been used on almost all new notebook computers and PCs since 2012. By the way, in everyday speech, BIOS is still often referred to (e.g., BIOS version, BIOS update, BIOS menu), even if the computer actually uses EFI or UEFI.

In this book, I assume in most chapters that you use a computer with (U)EFI. The reason I even go into the differentiation between BIOS and EFI here has to do with virtual machines: Although virtualization systems also support

EFI, a BIOS implementation is often used by default. This is because a parallel installation is unnecessary in virtual machines. In this respect, EFI offers hardly any advantages worth mentioning in this use case. So, the simpler and smaller BIOS is good enough for virtual machines.

2.2.1 EFI System Partition

For the parallel installation of multiple operating systems and their selection during the boot process to work, EFI requires a special partition (a reserved area) on the computer's SSD or hard disk. There, each operating system can install its startup program in its own directory (in technical jargon: its own *boot loader*). This partition is called the *EFI system partition* (ESP).

The EFI partition must contain a VFAT file system, which is a Windows 95–compatible file system. In addition, the partition must be specially marked.

When installing Linux, you must make sure that an EFI partition already specified by Windows is included in the `/boot/efi` directory. If no EFI partition exists yet, it must be created.

The installation programs of most Linux distributions take care of this step automatically—unless you decide to partition manually, in which case manual work and a little caution are required. Under no circumstances should an existing EFI partition be formatted: otherwise, none of the previously installed operating systems can be started!

ESP size

In the past, the partition was usually quite small (between 100 and 200 MiB). However, modern boot systems take up more space. In addition, ESP is also used for firmware updates. In the case of a parallel installation of Windows and Linux or of multiple Linux distributions, ESP quickly becomes too small.

For new installations of Windows 10/11, 512 MiB is now common. If you install Linux on an empty solid state drive (SSD) and the installation program gives you the option to freely choose the ESP size, you should use this guideline. It is almost impossible to increase the size at a later date. However, technically savvy users can set up a second EFI partition in an emergency (see *https://superuser.com/a/1231026*).

2.2.2 UEFI Secure Boot

UEFI Secure Boot is an extension of the EFI functions initiated by Microsoft. If Secure Boot is active, only an operating system that is signed with the key stored on the mainboard can be started. This prevents viruses or other

malware from interfering with the boot process. In the past, this system has been proven to have its flaws, but it has been retained.

A key from Microsoft

From a Linux point of view, however, this function causes problems: with Secure Boot active, Linux can only be installed and started if its boot program (more precisely: its boot loader) is signed with a key that exists on the mainboard. On most mainboards, there is only one key—namely, the one from Microsoft! Although Microsoft also provides the key to Linux distributors for a small fee, supporting Secure Boot has proven to be relatively difficult. There are currently two ways for Linux users to use Linux on computers with UEFI Secure Boot:

- **Using a Linux distribution that is compatible with (U)EFI Secure Boot**
  This is the case for the most popular Linux distributions, but not for the more exotic systems, which are not so widely used.

  Distributions with UEFI Secure Boot support use a boot loader signed with the Microsoft key, usually the Shim program. This program starts the usual GRUB Linux boot loader in a second step.
- **Disabling UEFI Secure Boot before installation**
  Fortunately, the EFI specification provides for this option. However, what this deactivation actually looks like is different on every computer or mainboard. If you can't find the option right away, you'll have to research on the internet how to change the setting on your computer.

  Note, however, that disabling Secure Boot causes problems with Windows 11. The security concept of Windows 11 has been tightened compared to that of Windows 10. Running Windows 11 without Secure Boot is technically possible, but it has limitations and is not recommended.

No Secure Boot without Signed Drivers

Secure Boot requires that the Linux kernel and all its modules are signed. However, this is only possible if all drivers run open-source code. This requirement is not met if you use proprietary drivers from NVIDIA or other manufacturers. In this case, you must disable Secure Boot in any case!

Additional information

You can read more details about EFI and the secure boot procedure on the following web pages:

- *https://en.wikipedia.org/wiki/UEFI*
- *https://wiki.archlinux.org/title/Unified_Extensible_Firmware_Interface*
- *https://help.ubuntu.com/community/UEFI*
- *https://rodsbooks.com/efi-bootloaders/index.html*

2.3 Installation Variants

There are various ways to install Linux. This section provides an initial overview of installation media (DVD, USB flash drive, network), installers (live system, text mode installer), and installation locations (internal/external media, parallel installation). From the next section on, this chapter then focuses on what most Linux beginners consider to be the "normal case": installing from a USB stick onto the SSD of their own notebook computer.

2.3.1 Installation Medium

ISO images

All Linux distributions offer ISO images for download on the internet. These images were originally intended to be used to burn a DVD and boot the computer with it. When installing Linux on a virtual machine, you can skip this step and simply use the ISO image as a virtual DVD drive.

USB flash drive

Because DVD drives have become a rarity on real computers, a USB flash drive has become the accepted installation medium. All ISO images with Linux installers are compatible with this method, so they can also be used on a USB flash drive as boot media.

But to write an ISO image to a USB flash drive, you need a special program. I recommend Etcher (see Figure 2.1) for this purpose, which you can download for Windows, macOS, and of course Linux from the following website: *https://www.balena.io/etcher*.

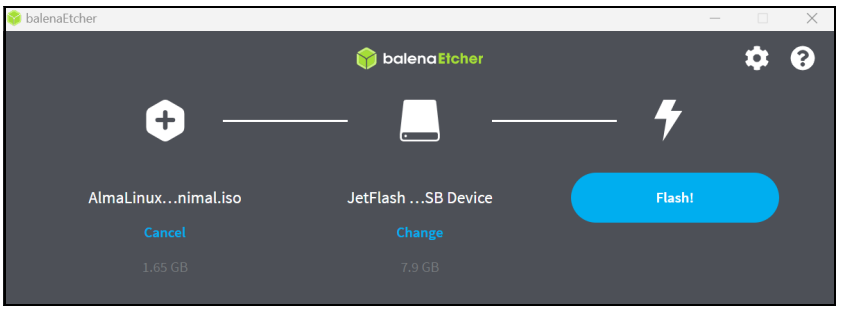


Figure 2.1 Etcher, Used to Transfer ISO Image to USB Flash Drive

Alternatively, some distributions offer their own programs for writing to USB flash drives. Fedora, for example, uses Fedora Media Writer. Linux professionals can also use the `dd` command instead of Etcher.

2.3.2 Network Installation

In a network installation, the installation files are read not from a USB flash drive, but from the network. There are two variants, which differ in how the installation starts:

- **Starting the installation using a USB flash drive**
  In this scenario, the installation is started from a USB flash drive. The installer helps to establish the network connection and then loads all other data from the internet or from the local network. This installation method is particularly popular with Debian using a netinst image.
- **Starting the installation through the network**
  This "real" network installation requires that your computer can load the boot data from the local network. Most common motherboards are capable of this if BIOS or EFI is set up correctly. This type of installation is useful when Linux is installed on many computers, such as in a school.

  To do this, there must be a server on the local network that offers the Linux installer in the form of boot data. However, setting up the installation server is not trivial. Only selected distributions support this installation method, including Red Hat and SUSE. If you want to install Debian on multiple computers automatically, visit the following page: *https://fai-project.org*.

Server and cloud installations

If you want to install Linux on a computer located in a remote data center, then different rules apply. The normal case is that the hosting or cloud company will take care of the installation for you. You just need to select the desired distribution and set a few parameters (e.g., for LVM and RAID configuration). A few minutes later, you can log in via SSH with a randomly generated password and start administrating the server. (SSH stands for Secure Shell. In simple terms, SSH is a tool for remote administration in text mode.)

Alternatively, some providers also allow you to carry out the installation manually. In this case, the installer will be launched by the hosting or cloud provider so that you can operate it in a web browser or virtual network computing (VNC) client.

2.3.3 Installation Program

The installation medium—regardless of whether it is read via DVD, USB flash drive, or the network—contains a complete Linux system. If the start of this system succeeds, this is already a good sign that the computer is Linux-compatible.

Installation in text mode

In the past, the installation media contained only a minimalist text-mode installer that is launched immediately, in addition to a basic Linux system. Some distributions (e.g., Arch Linux or Ubuntu Server) are still based on a text mode installation.

Installation in graphics mode

Installation programs that run in graphics mode and can be operated using the mouse or a trackpad are more convenient. Such installation programs are the norm today. All distribution installations presented in Chapter 3 work in this manner.

Installation program in a live system

Some distributions (such as Fedora and Ubuntu) have switched to placing a so-called "live system" on the installation medium—that is, a complete Linux system including desktop and basic configuration and application programs. The installer is just one program among many others.

This approach has two advantages: First, you can try out Linux much more thoroughly before installing it and check whether the graphics system and the connection to the local network work satisfactorily. Second, professionals can also perform repair work on an existing installation or recover data from a damaged computer in the live system.

2.3.4 Installation Location

Internal location versus data medium

Linux is typically installed on the internal SSD or hard disk of a notebook or desktop computer. A tempting alternative at first glance is to install Linux on an external USB drive. Unfortunately, with this installation variant, there are often problems booting the Linux system afterward. For this reason, this installation method is not recommended.

Linux alone or in parallel with Windows

The installation is especially easy on a new computer without an operating system or if you do not have to take existing data into account. Then you can simply specify in the installation program that you want to use the entire SSD for Linux.

However, usually Windows is already installed and supposed to continue to be used. In that case, you have to make room for Linux on Windows before you start the Linux installer.

Multiple distributions on one computer

To try out different distributions or to test a new version of your distribution in parallel with the existing version, you can install multiple distributions side by side on your hard disk or SSD. For this purpose, each distribution needs at least its own system partition, so the most important requirement is that there is space on your hard disk for more partitions.

2.4 Overview of the Installation Process

This section summarizes the steps of an ordinary Linux installation. *Ordinary* here means that Microsoft Windows is already installed on the machine. More details on these points will follow in the other sections ahead:

- **Downsize Windows partition**
  Normally, Windows fills most of the SSD in a single, very large partition. To make room for Linux, the size of this partition must be reduced. This step must be done in Windows. To do this, run **Create and Format Hard Disk Partitions** from the **Start** menu. To learn what a partition is and how much space you should free up, see Section 2.5.

  In the past, some Linux installers helped downsize Windows partitions. But even then, that rarely worked. If the Windows file system is encrypted with BitLocker, downsizing via Linux is completely impossible. For this reason, I strongly recommend that you perform this step directly in Windows.

- **Prepare USB flash drive**
  Also in Windows, you need to download the ISO image of your desired distribution and transfer it to a USB flash drive using Etcher or a comparable program. It is not enough to simply copy the ISO file to the USB drive as a file! The ISO file must be transferred in blocks. This will delete all the previous contents of the USB flash drive. (I have been using my own USB drive, only 8 GiB in size, for Linux installations for years.)

- **Start Linux installation**
  Now restart the computer with the USB flash drive plugged in. The installation system should start automatically. If Windows continues to boot instead of the Linux installation media, you must explicitly specify the boot media during computer startup. The required key combinations depend on the BIOS/EFI of your computer. On my test computers, for example, F8 or F12 does the trick.

  Some EFI versions place the USB drive *twice* in the selection list, once prefixed with *UEFI*. The two variants provide the option to use the USB drive in the old BIOS or the new UEFI mode. For a Linux installation, you definitely need the UEFI variant!

  If booting from the USB drive doesn't succeed, your BIOS/EFI is configured to not allow booting from external media. To change the BIOS/EFI settings, you must press a key immediately after switching on the computer, often Del or Enter or F1. If these keys remain ineffective, you

will have to research the correct key combination for your computer on the internet.

The installation program looks a little different for each distribution. For the most important distributions, concrete tips on how to use the installation program will follow in the next chapter. The first questions mostly concern the language of the user interface and the configuration of the keyboard and mouse.

- **Create Linux partitions**
  An essential step of any installation is the creation of Linux partitions on the SSD. Tips for optimal sizing of Linux partitions will follow in Section 2.7.

  Some distributions also allow you to specify whether the Linux file system should be encrypted and whether you want to use additional features such as LVM as part of the partitioning process (see Section 2.6).
- **Select installation scope**
  Some distributions allow you to select which parts of the Linux distribution you want to install. In other distributions, this step is omitted (e.g., with Ubuntu). Instead, a basic system is simply installed. You can then add further programs later as required during operation.

  In general, less is more during this phase of the installation! In the beginning, limit yourself to the programs you absolutely need and add more software only when you need it. This way you keep your system lean and minimize the effort of regular updates.
- **Configuration**
  Depending on the installation program, various configuration queries will follow now, such as which name and password the user account should use or under which host name the Linux system should appear in the network.
- **Boot loader**
  In the final step, the installation program sets up the so-called boot loader. This program (mostly GRUB) is later responsible for starting Linux. For EFI machines, the boot loader is installed in a directory of the ESP.

All in all, the initial installation of Linux will probably take about an hour. However, with a little practice and a fast computer, it can be done in 15 minutes. Afterward, you can start using Linux or manually perform further configuration steps and optimally adapt Linux to your special requirements.

2.5 Partitioning Basics

Before you downsize the Windows partition or decide in the installation program on the optimal size of the new partitions for Linux, you should know what a partition is and how it is used.

What are partitions?

Partitions are sections on the SSD or hard disk. Partitions with Windows file systems get their own letters (`C:`, `D:`, etc.) and seem to behave like independent hard disks.

Usually, there are already a few partitions on the data medium of a Windows notebook: the already mentioned ESP, plus possibly other small partitions with the boot system, drivers, and a recovery system, as well as a large partition that fills almost the entire SSD. This partition contains a file system with Windows and your personal data.

Why more partitions?

You will need additional partitions as soon as you want to install several operating systems on your computer at the same time. There are two reasons for this: On the one hand, different operating systems often use different *file systems*—that is, different methods of storing files within the partition. On the other hand, separate partitions avoid duplications and conflicts in directory and file names.

Linux usually provides for several partitions itself—for example, one partition for the operating system, another for your own data, and a third as a so-called swap partition. This is the counterpart to the Windows swap file.

For a Linux installation, it doesn't matter how much space is left on your SSD or hard disk on Windows. You cannot use this space—inside a Windows partition—for Linux. You need space *outside* the existing Windows partition(s) for the Linux installation to create new Linux partitions there.

Partitioning aids

To change the partitioning of the hard disk, each operating system provides its own tools:

- On Windows, there is the Disk Management program for this purpose. To run the program, search for "Create and format hard disk partitions" in the **Start** menu. To downsize the Windows partition, select it and execute the **Shrink Volume** context menu command.
- Distribution-specific partitioning aids are available during the Linux installation (see also Chapter 3).
- If you need to make changes to the partitioning after installing Linux, it is best to use the `parted` command or its graphical user interface, `gparted`.

Increased Flexibility with LVM

Changing the partitioning at a later date requires a great deal of effort. In many cases, the contents of a partition are lost when its size is changed. Moving partitions is also not provided for. For this reason, it is recommended to plan the partitioning well from the beginning.

Linux professionals can work around many limitations by using the LVM system (see Section 2.6). This is an intermediate layer between partitions and file systems.

MBR versus GPT

There are two methods for managing partitioning information on an SSD or hard disk:

- **GPT**
  The modern *GUID partition tables* (GPT) have been used in internal SSDs and hard disks in all computers since 2012.
- **MBR**
  Storing partition data in the *master boot record* (MBR) is outdated and has many limitations. Unfortunately, MBR is still used today, mainly in virtual machines with disks smaller than 2 TiB and with external disks (e.g., USB flash drives and SD cards).

2.5.1 MBR Basics

Primary, extended, and logical partitions

MBR is intended for disks up to a maximum of 2 TiB. There are three types of partitions: primary, extended, and logical. A maximum of four primary partitions can exist on the hard disk. It is also possible to define an extended partition instead of one of these four primary partitions. Inside the extended partition, you can then create multiple logical partitions.

The point of extended and logical partitions is to get around the historically imposed limit of only four primary partitions. Note that some partitioning tools do not differentiate between different partition types in the user interface and take care of how the partitions are created internally.

An extended partition serves only as a container for logical partitions. Only primary and logical partitions are suitable for the actual storage of data.

2.5.2 GPT Basics

All partitions are equal

With GPT, each partition is identified by a *globally unique identifier* (GUID). All partitions are equal; that is, there is no distinction between primary, extended, and logical partitions. Each partition can be up to 8 zettabytes (ZiB) in size: that's 2^{73} bytes, or about a billion TiB!

Note that the partition numbers do not have to match the actual order of the partitions. Suppose you create three partitions of 200 GiB each, then shrink the second partition to 50 GiB. This creates a gap between partitions 2 and 3 in which you can set up the fourth partition.

Comprehensive information about the structure of the GPT partition table and compatibility with various operating system versions can be found on the following Wikipedia page: *https://en.wikipedia.org/wiki/GUID_Partition_Table*.

2.5.3 Partition Names

Windows On Windows, partitions that the operating system can use are designated with drive letters. `A:` and `B:` are reserved for floppy disks for historical reasons. The other letters denote the partitions with Windows file systems.

Linux On Linux, internal access to hard disks or their partitions takes place via so-called device files (see Table 2.1). Hard disks and SATA SSDs are given the designation `/dev/sda`, `/dev/sdb`, `/dev/sdc`, and so on, in sequence. For SSDs for the NVMe interface, however, the device names start with `/dev/nvme`.

To address a single partition and not the entire hard disk, the partition number is added to the device name. With GPT partitioning, all partitions are simply numbered in sequence. In MBR partitioning, on the other hand, the numbers from 1 to 4 are reserved for primary and extended partitions. Logical partitions always start with the number 5.

Device Name	Meaning
`/dev/sda`	First SSD/hard disk
`/dev/sdb`	Second SSD/hard disk
`/dev/sda1`	First partition of SSD/hard disk `/dev/sda`
`/dev/sda2`	Second partition
`/dev/sda5`	First logical partition (MBR only)
`/dev/sda8`	Fourth logical partition (MBR only)
`/dev/nvme0n1`	First NVMe SSD
`/dev/nvme0n1p1`	First partition
`/dev/nvme0n1p2`	Second partition

Table 2.1 Some Sample Device Names of Partitions

2.5.4 File Systems

Partitioning only reserves space on the SSD or hard disk. Before you can store files in a partition, you must create a so-called file system. It contains administrative information in addition to the actual data. Creating a file system in a partition is also called *formatting*. Windows and Linux use different file system types:

- NTFS is commonly used on Windows. On small USB flash drives and SD cards, the rather old VFAT system is often used instead, and on larger SD cards, the exFAT file system is often used.
- On Linux, `ext4` is the most popular file system type. Alternatives are `xfs` (ideal for huge file systems) and `btrfs`.

2.6 LVM and Encryption

This section introduces the basics of LVM and encryption. As a matter of fact, LVM is a fairly advanced topic that is out of place at this point in the book. However, especially for notebook computers, the encryption of data media is desirable—and most Linux distributions only offer that in combination with LVM. In addition, the decision for or against encryption—unlike on Windows or macOS—cannot be revised later. Therefore, I recommend that you at least skim this section so that you can better appreciate the scope of these options.

2.6.1 Logical Volume Manager

The *Logical Volume Manager* (LVM) puts a logical layer between partitions and file systems. What sounds very abstract at first has quite tangible advantages in practice:

- In the section managed by LVM, you can create, resize, and shrink virtual partitions (strictly speaking: *logical volumes*) at runtime. You can increase the existing LVM storage pool at any time by installing another hard disk or SSD.
- You can also combine sections of multiple disks into a single, huge storage pool thanks to LVM.
- You can create snapshots. This is useful for carrying out backups during operation.
- Despite these features, the speed difference compared to directly addressing a hard disk partition is hardly measurable. So you don't pay with performance for the flexibility you gain.

Glossary

Several similar terms and abbreviations make it difficult to get started in the LVM world. There are three levels between the hard disk and the file system: physical volumes, volume groups, and logical volumes:

- **Physical volume**
  A physical volume is usually a partition of the hard disk or SSD managed by the LVM. For the partition to be used as a physical volume, it must be specially marked.
- **Volume group**
  One or more physical volumes can be combined into a group. In this way, it is possible to virtually concatenate partitions of different hard disks—that is, to use them uniformly. The volume group represents a kind of storage pool that combines all available physical storage media.

  In a new installation of Linux, the pool consists of only one physical volume; in this respect, it is not a real group at all. The concept remains valid, however, because the pool can be expanded later with additional physical volumes if necessary.
- **Logical volume**
  A logical volume is a part of the volume group. For the user, a logical volume acts like a virtual partition. The file system is created in the logical volume. If a logical volume later turns out to be too small, it is relatively easy to enlarge it retroactively (provided there is still free space in the storage pool—i.e., in the volume group).

The installation program takes care of designating a partition as a physical volume, creating a volume group from it, and setting up logical volumes in it—usually one each for the root file system and for swap space, and possibly another for `/home`.

The extent to which you are confronted with the three terms just defined depends heavily on the installation program.

2.6.2 Encryption

Many distributions provide an option to encrypt the file system during installation. A password must then be specified at system startup before the file system can be accessed. Provided that you use a sufficiently long and complex password that cannot be guessed easily, encryption effectively protects your data: even if your notebook computer falls into the wrong hands, no one will be able to read your files.

What does encryption have to do with LVM? Most encryption systems are based on the fact that the file system is addressed not directly, but via an

intermediate layer that is responsible for the encryption. Technically speaking, the procedure is very similar to that of LVM.

Many installation programs therefore offer encryption functions only in combination with LVM. Behind the scenes, not a single file system, but the entire LVM layer is encrypted.

2.6.3 Limitations

The use of LVM and encryption doesn't only have advantages. It's also associated with restrictions:

- LVM administration is relatively complicated. During installation, the installer will assist you in setting up LVM or encryption. However, if you then want to change the configuration while the system is running, you will have to rely on relatively bulky commands in most distributions. For detailed information on how to use these commands, see Chapter 18.
- If you use encryption, the password must be entered manually every time you start the computer. Basically, this approach is unsuitable for servers that are not located on site.
- For a subsequent reinstallation of Linux, such as when you change over to another distribution, it's very difficult and in many cases impossible (this is especially true for Ubuntu and all derivatives) to take over an existing encrypted data partition. Most of the time, you're forced to make a complete backup first and then restore it in the newly installed Linux system.

2.6.4 Recommendation

If you're starting out with Linux and performing your very first installation, I would advise you not to use LVM and encryption because of the complexity involved. As a rule, it's better to gain some experience with Linux first (quite possibly in virtual machines).

If you're sure that you want to use Linux on your notebook computer permanently, and as soon as questions like "Which distribution?" or "Which layout of the SSD?" are finally decided, there is really no way around encryption (and thus also LVM). This is the only way to prevent not only your device but also all your data from changing hands if your notebook computer is lost or stolen. Complete encryption should be a matter of course for notebook computers that are used for business purposes.

2.7 Creating Linux Partitions

One of the most important steps during Linux installation is the creation of new Linux partitions. All common installation programs contain easy-to-use partitioning tools for this purpose. Figure 2.2 shows the partition editor of Linux Mint, executed on a test computer on which Ubuntu has already been installed before.

At this point, we are dealing with fundamental issues: How many partitions should you set up for Linux? In what size? What are the implications for speed, for subsequent maintenance, and for a possible reinstallation of a different or updated Linux distribution?

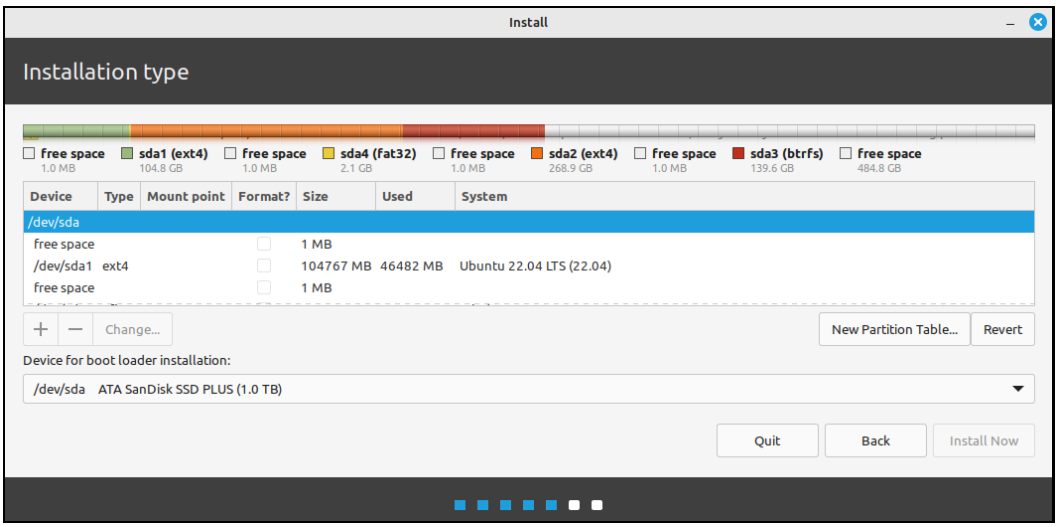


Figure 2.2 Linux Mint Partition Editor

Powers of Two versus Powers of Ten

In this book, in most other Linux documentation, and for most Linux tools, bytes are counted in powers of two:

1 KiB = 2^{10} bytes = 1024 bytes
1 MiB = 2^{20} bytes = 1024^2 bytes = 1,048,576 bytes
1 GiB = 2^{30} bytes = 1024^3 bytes = 1,073,741,824 bytes
1 TiB = 2^{40} bytes = 1024^4 bytes = 1,099,511,627,776 bytes

Many SSD manufacturers and also some file managers calculate decimally—that is, with powers of 10: 1 TB = 1 TB = 10^{12} bytes = 1,000,000,000,000 bytes. Thus, an SSD that is 1 TB according to the manufacturer only holds about 931 GiB according to the conventions in this book.

2.7.1 Number and Size of Linux Partitions

Time and again people ask me how an SSD with *n* GiB should best be divided into partitions. Unfortunately, there is no universal answer to this question. However, this section should at least give you some rules of thumb for the right number and size of partitions.

System partition

The system partition is the only partition you absolutely need. It accommodates the Linux system with all its programs. This partition is always used on Linux via the / directory. This location denotes the root—that is, the beginning of the directory tree. For this reason, the system partition is often referred to as the *root partition*.

A reasonable size for installing and running a common distribution for desktop use is 30 to 40 GiB. Then, of course, there's the space required for your own data—unless you store your files in a separate home partition.

Boot partition

In the past, it was necessary to create a separate partition named `/boot` for technical reasons. This partition houses the files that are needed during the first phase of the computer startup: in particular, the `vmlinuz*` kernel file and the `initrd*` initial RAM disk file.

Since EFI and the GRUB 2 boot loader have become established, the boot files can be read directly from the system partition even in complex LVM and RAID setups. A separate boot partition is then actually superfluous. However, some distributions (RHEL) still suggest setting up this partition. You can follow this suggestion; even if the partition is not required, it doesn't cause any damage. If you decide to have your own boot partition, it should be around 1 GiB in size.

Home partition

A home partition separates the storage locations for the system files and your own files. This has a significant advantage: you can easily install a new distribution into the system partition later without compromising the separate data partition containing your own data.

For the home partition, `/home` is used as the `mount` directory. It isn't possible to give a recommendation for the size of this partition because it depends too much on what tasks you want to do with your Linux system. If you are sure that you do not want to install any more operating systems on your computer, you can make the home partition so large that it fills the entire remaining space on the SSD.

More data partitions

The division of the hard disk into partitions can be pushed even further. For example, if you want to use the Linux computer within a larger network as a special server for network or database tasks, you can provide separate partitions for the resulting data and select a file system that is optimal for the

type of data access. However, this type of optimization is only useful for Linux experts.

Provided that there is still unpartitioned space on your hard disk, it's no problem to add more partitions to a running system and, if necessary, to move data from an existing partition to a new one. So if you are unsure, simply wait a little longer before partitioning and leave part of the hard disk without partitions.

Swap partition

The swap partition is the counterpart of the Windows swap file: if there isn't enough RAM available to Linux, it stores some of the RAM content that isn't currently required in that partition. Unlike the other partitions, the swap partition doesn't get a name (no `mount` point). The use of a separate partition instead of an ordinary file as on Windows has mainly speed advantages.

Having 2 GiB of space for the swap partition is sufficient in almost all cases. More and more installations are completely abandoning the swap partition. In this case, Fedora just compresses unused data blocks in the RAM in order to use the main memory particularly efficiently. (In the past, the advice was to make the swap partition at least as large as the main memory so that a notebook computer could be put into hibernation mode. However, hibernation is no longer supported by most distributions. There is only the standby mode, in which the notebook continues to use some energy.)

Swap File on Ubuntu

Ubuntu no longer sets up a swap partition by default for desktop installations. Instead, the installer provides for a swap file, similar to Windows or macOS. Although this is slightly slower, it offers more flexibility if the swap file is to be reduced or enlarged later. It also saves a partition that is difficult to change later.

EFI system partition

On EFI systems, there must be at least *one* ESP. This partition is set up by default when the first operating system is installed, whether it is Windows or Linux. If additional operating systems are installed later, all operating systems share the common EFI partition, and each places files there to start the operating system. The ESP must contain a VFAT file system and is accessed via the `/boot/efi` directory on Linux. The ESP should be at least 512 MiB in size.

Conclusion

With any Linux installation, you need a system partition. Setting up additional partitions is optional, strongly dependent on the intended use of

Linux, and also a matter of taste. My personal recommendations for a first-time Linux installation is summarized in Table 2.2.

Directory	File System Type	Usage
/boot/efi	VFAT	ESP (at least 500 MiB)
	–	Swap partition (2 GiB)
/	ext4	System partition (30 to 50 GiB)
/home	ext4 or xfs	Own data (size as needed)

Table 2.2 Recommended Partitions for Desktop Use

2.7.2 Which File System?

Linux supports a lot of different file systems, including ext4, btrfs, and xfs. These file systems are presented in detail in Chapter 18.

ext4

The most popular file system type for Linux is ext4. This file system is well developed and has been robust and efficient for many years.

xfs

The xfs file system is the default for RHEL. It is especially well suited for servers with multiple-TiB-sized file systems.

btrfs

The btrfs option provides more features than any other Linux file system. At the same time, however, its many features complicate the administration. Notwithstanding, Fedora and SUSE use btrfs as the default file system for the system partition. Red Hat, on the other hand, is of the exact opposite opinion: the company has designated the file system as *deprecated* in 2017. Therefore, btrfs is not supported in RHEL.

I strongly advise desktop users not to use btrfs! Doing so requires a lot of administration know-how, especially in the SUSE default setup, and easily leads to unexpected problems. At best, the btrfs advantages outweigh the disadvantages for server installations that are managed by experienced administrators. The simplicity and stability of ext4 and xfs are more important arguments for me than all the extra features of btrfs combined.

Swap partition

No real file system is set up in the swap partition. However, the partition must be formatted by mkswap before it's used for the first time. All Linux distributions take care of this automatically.

Recommendation

Table 2.2 summarizes which file systems you are best off using for which partitions. If there is an additional boot partition, an ext4 file system is suitable for this.

2.8 Setting the Scope of the Installation

Some distributions allow you to select which components, programs, or packages are installed during installation (see Figure 2.3). Software packages are often selected as preconfigured groups. Other distributions simply install a basic system without any choice. If necessary, you can install all other programs during operation.

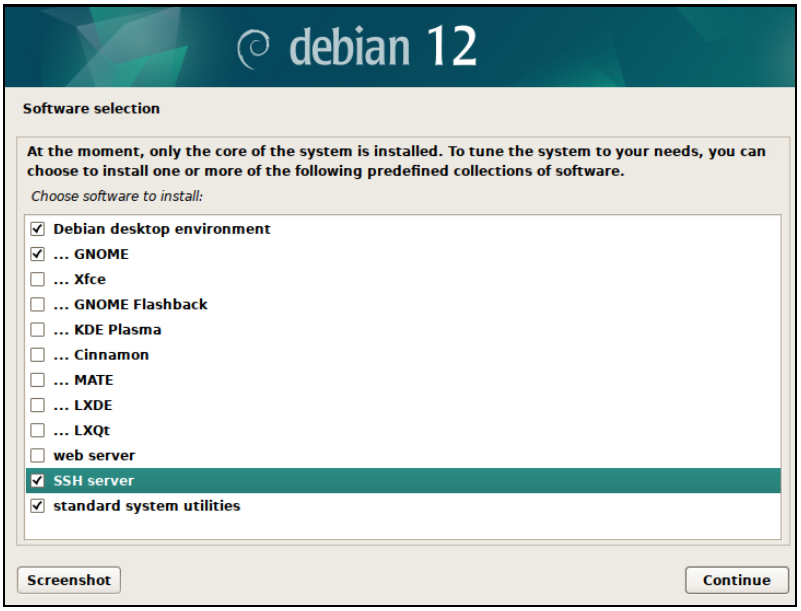


Figure 2.3 Setting Debian Installation Scope

GNOME, KDE, or another desktop system

Some distributions allow you to choose between desktop systems like GNOME, KDE, Cinnamon, MATE, or Xfce. These are different user interfaces for Linux. Each desktop system has specific advantages and disadvantages. GNOME is easy to use, but KDE provides more settings for technically experienced users. If you are a Linux newbie and unsure, you should choose GNOME! This desktop system is currently the most widespread.

Some other distributions lack a selection option in the installation program, but you can decide on a certain desktop variant or a spin (Fedora) when downloading the ISO image.

2.9 Basic Configuration

This section provides some background information on the common steps of the basic configuration. The order, details, and scope of the basic configuration vary greatly depending on the distribution. Some distributions

keep the configuration to a minimum during installation. Further hardware configuration is then only carried out in the running basic system. In general, the following applies: almost all settings can also be made later. Simply postpone the configuration of components currently not needed until later!

Root password

On Linux, the `root` user is usually responsible for system administration. This user has unrestricted rights, but of course this also means that the potential for damage is unrestricted. It is therefore essential that access to root be secured with a strong password.

In Ubuntu and some other distributions, the `root` user account is completely disabled. Password protection for `root` is not necessary there. Administrative tasks are performed by designated users on Ubuntu and require the user password to be entered again.

In Debian, it's permissible to skip setting the `root` password. The `root` login is then locked, and the first created account is responsible for administration, as with Ubuntu.

In openSUSE, `root` and the default user are given the same password. If you don't want that to happen, you need to disable the easy-to-miss option in the installer.

User administration

It is unusual on Linux to work as `root`—except, of course, when performing admin tasks. When you write an email, compile a program, or browse the internet, you should log in as an ordinary user. During installation, you have the option to set up one or more such users, including their passwords. Later, you can add more users, change the passwords of existing users, and so on.

A user name (account name) is composed of lowercase letters. Spaces are not allowed, and special language characters should be avoided.

The password should be at least eight characters long. Use a secure password! However, do not use special language characters (e.g., äöüß) or other letters that are not defined in the ASCII character set. Such characters could result in you being unable to log in if the keyboard is configured incorrectly or not configured at all.

Automatic network configuration

The network configuration will be done automatically if the installation program detects a DHCP server in the local network. This task is performed by a router. In this case, the network configuration is reduced to specifying the desired computer name and, if necessary, the Wi-Fi password.

Host and domain name

The hostname is simply used to define the computer name. Do not use `localhost` as the hostname! This name has a special meaning.

You can choose the domain name freely at home. For company installations, it is predefined and often corresponds to the workgroup name used on Windows.

Firewall

Some distributions protect network or internet access with a firewall by default. This firewall allows connections initiated by you, but blocks requests coming from the outside, thus significantly increasing security. If you intend to offer network services on your computer yourself (e.g., an SSH or web server), you should define exceptions for these services.

Installing the boot loader

The boot loader determines how Linux will be booted in the future. The GRUB program is responsible for this in almost all distributions. It is installed automatically; only a few distributions offer configuration options at this point. If you do need to specify a location for the boot manager, it's usually `/dev/sda` or `/dev/nvme0n`, which is the first hard disk or SSD.

First start

When rebooting after installation, the freshly installed Linux system is automatically booted. After that, you will begin your first exploratory journey through the Linux world. If you want to boot Windows instead of Linux, you need to press a special key combination during reboot to open the EFI boot menu.

2.10 System Changes, Extensions, and Updates

Once your Linux system is running stably, you will mostly want to configure, extend, or update it according to your own ideas.

Detailed information on these topics is spread throughout the book. This section is therefore primarily intended for reference, to save you the search work as much as possible.

2.10.1 Software Installation and Package Management

Depending on the distribution, there are various commands and programs that you can use to install, update, or remove additional software packages during operation. In most GNOME-based distributions, the software program is responsible for this. The underlying procedures and commands that you can use to install software in the terminal are described in Chapter 16.

2.10.2 Updates

Normal updates

All common distributions notify you of updates regularly and automatically. With a few mouse clicks, you can update all affected programs. Unlike

Windows, the update system is responsible for *all* components. So there are not different update mechanisms for different programs.

Through the update system, serious bugs are fixed and security updates are performed. The first update after reinstalling a distribution often takes a very long time, sometimes longer than the actual installation! This is because it installs all updates that have been released since the distribution was completed. All further updates, which are performed on a regular basis, then only affect a few packages and are performed correspondingly faster. Such updates are quite frequent: They usually appear weekly, sometimes even several times a week.

Distribution updates

The "ordinary" update system fixes bugs and security flaws, but usually does not install fundamentally new program versions. (The only exception is for web browsers.) So you will wait in vain for an update from LibreOffice 7.*n* to version 8.*n*, even if the latest LibreOffice version has already been released. Instead, you are mostly forced to update the entire distribution to the next version—hence the term *distribution update*.

There are two different procedures for distribution updates: either you start the installation from a disk and then specify that you want to update an existing distribution, or you perform the update at runtime and then only need to reboot. The second method is more elegant because it can be performed without installation media. The new packages are simply downloaded from the internet. It also minimizes the time during which the distribution is not running or during which a server is offline.

Table 2.3 summarizes which distributions support which procedures. Note that with RHEL, you *automatically* receive all updates within a major release over a period of many years. Your RHEL version thus increases, for example, from 8.0 to perhaps 8.9. However, an update to the next major version, such as from 8.9 to 9.0, is not provided for. This would require a new installation.

Distribution	Update during Installation	Update during Operation
RHEL		•
Debian		•
Fedora	•	•
openSUSE	•	•
Ubuntu	Only up to version 22.10	•

Table 2.3 Methods for Distribution Updates

Problems

What sounds great in theory unfortunately often works poorly in practice. After the distribution update, sometimes programs do not work as before, and searching for the errors can be time-consuming. I myself have lost faith in distribution updates after countless problems.

Personally, I tend not to go along with every distribution update unless the work on this book forces me to. Instead, I perform a complete reinstallation when necessary—often after three or four years—leaving only the `/home` data partition unchanged.

Rolling releases

Rolling releases are intended to eliminate the need for distribution updates altogether. In distributions that follow the rolling release model, all packages are constantly updated to the latest available version—as is already done with web browsers.

This concept also sounds better than it actually works: many innovations inevitably lead to incompatibilities or migration problems. Automatic updates may occur at an inopportune time, and the user is suddenly confronted with programs that no longer function as before—or not at all.

For these reasons, rolling release distributions have not yet been able to establish themselves or are aimed exclusively at technically experienced Linux professionals:

- Debian comes pretty close to the rolling release model if you enable the *testing* or *unstable* package sources.
- SUSE offers *Tumbleweed*, a rolling release version of openSUSE that has been working quite smoothly since 2014.
- The Neon KDE distribution partially follows the rolling release model. Although the substructure, an Ubuntu LTS version, remains unchanged during updates, the KDE packages are constantly updated to the latest version.
- Arch Linux and related distributions such as Manjaro also rely on the rolling release model. I switched my main machine to Arch Linux in 2022 and have been very happy with it so far. But of course, I'm a prime example of the target audience for rolling release distributions.

2.10.3 Configuration

Although there have been repeated efforts in the past to unify the configuration of Linux, in fact the individual distributions unfortunately still differ considerably. For this reason, you should use the respective supplied tools for further configuration if possible.

Admittedly, solving some configuration problems requires more than a few mouse clicks. That's why in this book I go into detail about the basics and background of various software and hardware components, independently of specific distributions (see Table 2.4).

Topic	Chapter	Topic	Chapter
GNOME	Chapter 4	System startup	Chapter 19
KDE	Chapter 5	Kernels and modules	Chapter 21
Basic configuration	Chapter 14	Network configuration	Chapter 15
Package management	Chapter 16	Server configuration	Chapter 22 onwards
Graphics system (X and Wayland)	Chapter 17		

Table 2.4 Linux Configuration

Chapter 3
Installation Instructions

The previous chapter covered the basics of a Linux installation in detail, and now this chapter describes concrete examples. You'll get to know some selected Linux distributions in more detail and learn which special features need to be considered when installing them. At the same time, this chapter provides tips for the first steps after the installation.

The following distributions are described:

- Debian
- Fedora
- Linux Mint
- Manjaro
- openSUSE
- Pop!_OS
- Ubuntu

Server Distributions

Instructions for server installations will follow in Chapter 22, where I describe the installation of AlmaLinux, Debian Server, Oracle Linux, RHEL, Rocky Linux, and Ubuntu Server and go into the details of a server-specific installation, such as RAID and LVM configurations.

Recommendations for desktop use

For Linux beginners, I recommend Ubuntu. This is probably the widest distribution among home users, and in case of problems you can consult numerous forums and wikis.

A good alternative is Linux Mint. This distribution is considered even more beginner-friendly than Ubuntu. Mint has the additional advantage that it doesn't use the Ubuntu-specific Snap packages.

Fedora also rarely causes problems in desktop use, but it's explicitly intended for advanced Linux users. Red Hat views Fedora as an experimental platform. In this respect, Fedora is ideally suited for getting to know the latest developments from the Linux world. The biggest disadvantage of Fedora is its short update period. After one year at the latest, you must perform a distribution update or a new installation.

I used to mention openSUSE at this point, but the currently available version 15.5 is based on SUSE Enterprise 15 and contains some very old software components. The most attractive openSUSE version in my view is the rolling release variant, Tumbleweed.

Many years of updates

An important criterion for the choice of a distribution is its maintenance period: if you want to use Linux without any problems for a number of years, you should choose an LTS version of Ubuntu (20.04, 22.04, etc.) or a RHEL clone (e.g., AlmaLinux). These distributions each offer an update guarantee for at least five years.

This is perfect if you want to install Linux on the computers of your friends and relatives! With some restrictions, the update guarantee also applies to some Ubuntu-derived distributions like Linux Mint or Pop!_OS LTS.

3.1 Debian

Debian: "pure" Linux

No distribution represents "pure" Linux as much as Debian—and that's for several reasons:

- Debian is developed exclusively by a free community of developers. Behind Debian there is neither a company nor commercial interests, but, according to Wikipedia, more than 1,000 developers, most of them working for Debian on a voluntary basis. As a logical consequence, both Debian itself and access to its updates are completely free.
- One of Debian's central goals is that the distribution remains truly "free" in the sense of the open-source idea. The integration of binary drivers or commercial software without freely available source code is taboo. The only exception is firmware files for hardware devices, which have been included by default since version 12, although there is no open-source code for them.
- With Debian, stability takes precedence over very recent versions. That is why an ordinary Debian installation always lags a little behind the current state of development for many important components (kernel, GNOME, server services, etc.). If you need more recent versions, you can install them from the *testing* or *unstable* package sources.
- Debian supports many more hardware platforms than any other distribution. This is another reason why the development of a new Debian version often takes longer than planned.
- The Debian Project is governed by a democratic organization whose leadership members are elected on a regular basis. The rules of the game are formulated in a "social contract" (see *https://www.debian.org/social_contract.1.0.en.html*). This social contract contains the Debian Free

Software Guidelines (DFSG). The guidelines formulate the requirements a software project must meet in order to become part of the official Debian packages.

Debian's importance extends far beyond what can be measured in market share: Debian is an important and indispensable foundation for numerous other distributions, first and foremost for Ubuntu. Many Debian tools, starting with package management, have found their way into other distributions.

Desktop and server use

Debian cannot be clearly classified as a desktop or server distribution; thanks to its universal orientation, Debian is suitable for both purposes. I will go into more detail about some of the special features of server installation in Chapter 22, Section 22.4.

Code names

Debian uses a code name for each version, and each name matches that of a character from the *Toy Story* movie (see Table 3.1). In this book, references to Debian are to Debian 12 unless I explicitly give a different version number.

Code Name	Version	Completion
Buster	Debian 10	July 2019
Bullseye	Debian 11	August 2021
Bookworm	Debian 12	June 2023
Trixie	Debian 13	Expected 2025

Table 3.1 Debian Versions

Versions/variants

Compared to other distributions, Debian doesn't have countless distribution variants. There is only *one* Debian, which consists of a pool of about 64,000 packages. The exact number varies depending on the CPU architecture.

The basic system can be installed either from a DVD image or from a network installation image (`netinst image`, around 350 MiB). This image contains only the installation program. All packages are downloaded from the internet or from a local server during installation.

The hardware support is impressive. While other distributions usually support only two or three CPU platforms, Debian Buster supports 10: besides standard PCs (`amd64` and `i386`), these are ARM (`armel`, `armhf`, and `arm64`), MIPS (`mips`, `mipsel`, and `mips64el`), PowerPC (`ppc64el`), and S390. For some platforms, there are not only installation images, but also live images.

Additional information

Comprehensive information about Debian can be found on the official website:

- *https://www.debian.org*
- *https://www.debian.org/releases/bookworm/amd64*
- *https://www.debian.org/releases/bookworm/amd64/release-notes*

3.1.1 Installing Debian

With Debian, you can use two fundamentally different installation variants: On the one hand, you can write a live image to a USB flash drive and run the very user-friendly installation program included on it. On the other hand, you can use the traditional installation media, which includes the tried and tested installation program. It supports more configuration variants but is much more cumbersome to use and unsuitable for beginners.

Firmware files

Up to and including Debian 11, the "common" images lacked firmware files, which are not available as open-source code. For a modern notebook, this can result in your computer not even being able to establish a network connection. Fortunately, there were modified images *with* the firmware files for such cases.

With version 12, Debian has fortunately opted for a more pragmatic approach and now generally includes the firmware files. This eliminates the confusing differentiation between images that contained only open-source software and images that were simply required for many computers.

3.1.2 Live-Image Installation

Debian provides live images for various desktop systems to choose from (GNOME, KDE, Xfce, etc.). These variants are important because, unlike the conventional installer, they do not allow you to influence the scope of the installation. Thus, the desktop system that runs in the live system is always installed (see *https://cdimage.debian.org/debian-cd/current-live/amd64/iso-hybrid*).

As usual, start the installation by restarting your computer and plugging in the USB flash drive onto which you previously transferred the image. After a few seconds, you will enter the desktop of the live system, where you can first select the keyboard layout and then run **Activities • Install Debian**. When starting the installer, you will be asked for a password. It is `live`. After the start of the installation program, you can select a different language.

Things get exciting in the fourth step, where you decide on the partitioning of your hard disk or SSD (see Figure 3.1). In many cases, the **Install alongside**

option is the right one: if you use this option, the installation program shrinks an existing partition and sets up the partitions required for Debian in the space that's now free.

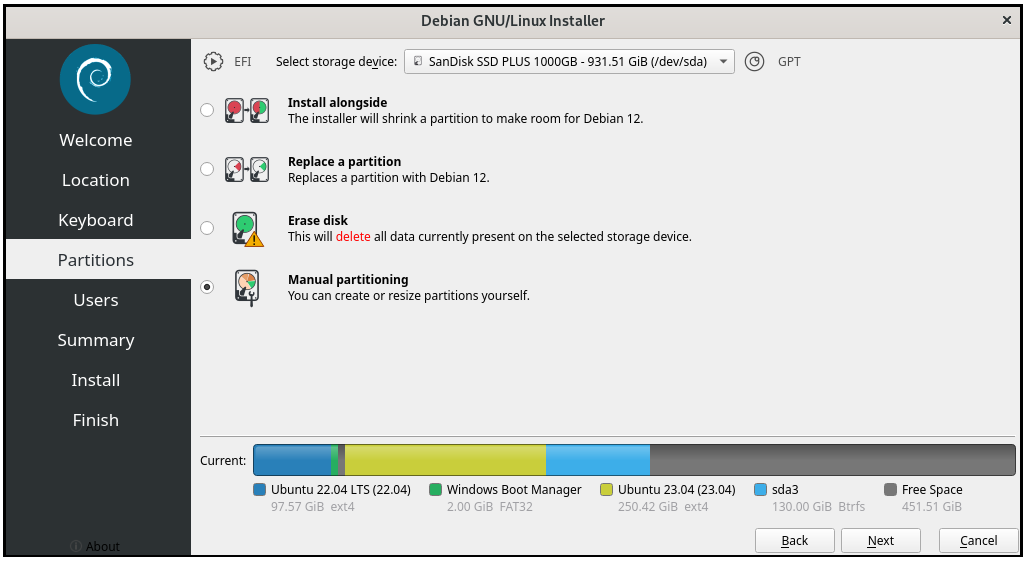


Figure 3.1 Installation Program Provides Several Partitioning Variants to Choose Among

Partitioning

If necessary, you can opt for **Manual partitioning**. This will give you the option in the next step to set up the desired partitions in the free area of the hard disk or SSD itself, select the file system type you want, and specify the directory (/ for the root partition, `/boot/efi` for the EFI partition, etc.).

Linux account

In the next step, you enter your name and the desired password for your Debian account. Deviating from the usual Debian customs, this user is given `sudo` rights; that is, it is allowed to perform administration tasks later on.

Summary

Before the actual start of the installation, the program displays a summary of all settings that have been made (see Figure 3.2).

Calamares Installer

The *Calamares installer* on the live images is not a Debian in-house development, but comes from a cross-distribution project (see *https://calamares.io/about*).

The installer can be adapted relatively easily to the requirements of the respective distribution. It is also used by various Ubuntu derivatives, by Linux Mint, by Manjaro, and by KDE Neon.

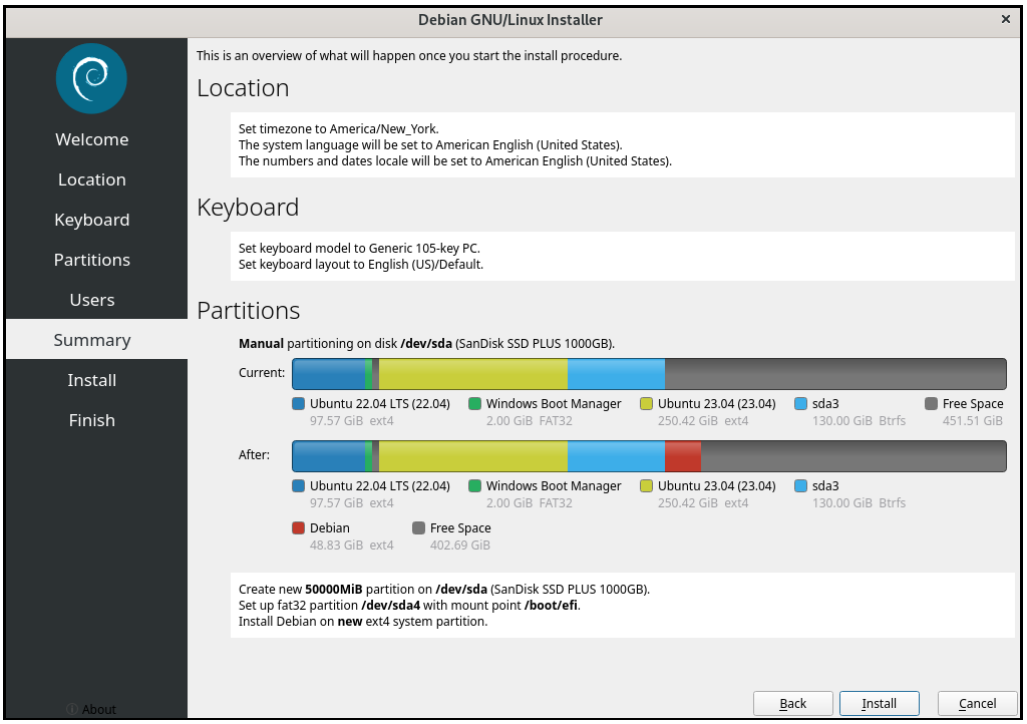


Figure 3.2 Summary before Starting Actual Installation

3.1.3 Standard Installation

Advanced Linux users who want to control the details of partitioning or the scope of installation more precisely are better off using the conventional installation method. You can download suitable images from the following website: *https://debian.org/CD*.

A menu is displayed at the beginning of the installation. By default, the installation program starts in graphics mode. If hardware problems occur, press **Install** to start the installation in text mode or run **Advanced Options • Expert Install**. Then you can have a very precise influence on the individual installation steps and, in particular, on the loading of kernel modules.

root password and user

In the first steps, you will then set the language and keyboard layout. In the dialogs that follow, you need to enter the password for `root` and create a new user.

Automatic sudo Configuration if root Password Is Empty

You can leave the `root` password blank and just click **Next**. In this case, the `root` login remains locked. However, the first user will then be given admin rights, as with Ubuntu, and can switch to `root` mode via `sudo`.

Partitioning the hard disk

The installation program provides the following options for partitioning the hard disk, among others:

- **Guided—Use the Largest Continuous Free Space**
  This option only displays if there is free space on the SSD that is not used by partitions. The installation program then sets up the partitions required for Debian in this area.
- **Guided—Use Entire Disk**
  The installer deletes all partitions and then uses the entire hard disk or SSD for the Debian installation. Another dialog will appear later asking whether you want to store all data in one partition, whether you want a separate home partition (this is recommended), or whether separate partitions should also be created for the `/usr`, `/var`, and `/tmp` directories. The latter is rarely useful.
- **Guided—Use Entire Disk and Set Up LVM**
  As the previous option, but with an LVM system that provides greater flexibility for later changes.
- **Guided—Use Entire Disk and Set Up Encrypted LVM**
  As the previous option, but the LVM system is encrypted. The key must be specified for each boot process; that is, this variant is only conditionally suitable for server installations.
- **Manual**
  This item enables you to perform the partitioning process yourself. However, you can also choose one of the other variants and change the installer's suggestions according to your own ideas.

Regardless of which variant you choose, you must explicitly confirm the partitioning plan again, so there is no risk that the installation program will make the partitioning hastily and irrevocably.

When you perform the partitioning manually, the installation program displays a list of all available partitions (see Figure 3.3). Select existing partitions by double-clicking or pressing Enter. Create new partitions by clicking the **Free Space** item at the bottom of the list. You can also shrink existing Windows and Linux partitions to make more room for new Linux partitions.

Unfortunately, the nested dialogs for editing the partitions are confusing and make little use of the capabilities of a graphical user interface. Many texts in the dialogs are to be interpreted as menu commands and when you click them, they lead to further dialogs. For example, clicking the **Use As: Do Not Use partition** line opens a dropdown list where you specify the desired file system type.

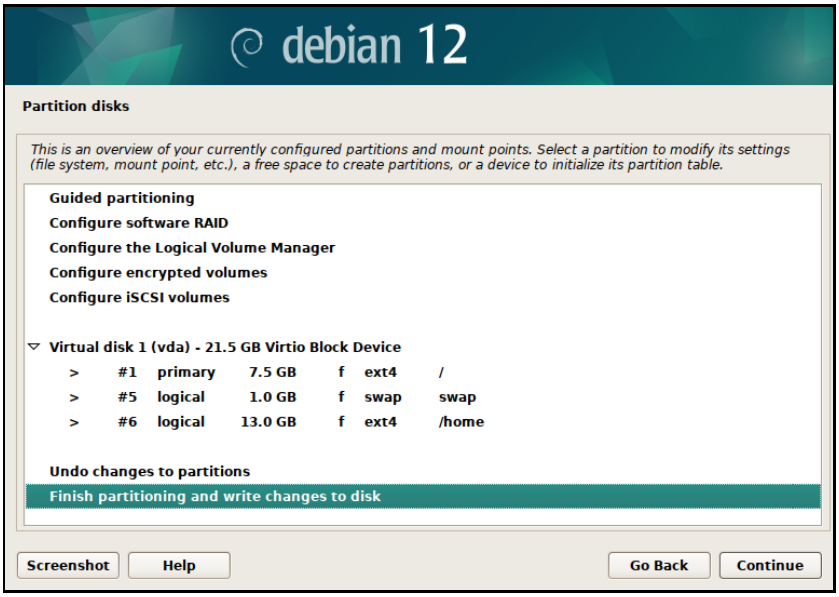


Figure 3.3 Partitioning Hard Disk/SSD

With **Done setting up the partition**, you can save the settings of the last edited partition. After that you can edit another partition or finish the partitioning process by clicking the **Finish Partitioning and write changes to disk** entry below the partition list. The installation program displays a summary of the planned changes to the hard disk partitioning and finally executes them after another confirmation.

If the SSD or hard disk was previously unused, the installer will set up a partition table on its own. On EFI machines, Debian opts for the GPT format. In virtual machines without EFI, Debian stores the partition table in the master boot record (MBR). The installation program doesn't offer any choices in this regard.

Configuring the package manager

In the dialogs that follow, the installation program asks if there are additional installation media (DVDs or additional USB flash drives; there usually are not) or if it should use a mirror server. From this server it can then download packages that are not included in the installation image. The mirror server also serves as a source for updates. Answer **Yes** and select a geographically close server in the subsequent step.

After installing some packages, you will be asked if you want your package selection to be reported to a central server so that the Debian developers can determine the most popular packages.

In the next dialog, you first perform a software selection: the **Debian Desktop Environment** package groups for various desktop systems as well as

Web Server, **SSH Server**, and **Standard System Utilities** are available for selection (shown in Figure 2.3 in Chapter 2).

You can even select multiple desktop systems at the same time in this dialog. Then, every time you log in, you can choose which desktop system you want to use. At the same time, however, choosing multiple desktop systems results in an unnecessarily bloated system with a lot of redundant software, such as multiple audio players, imaging programs, and so on. For this reason, multiple selection is not recommended at this point.

3.1.4 Getting Started

sudo versus su

Depending on how you performed the installation, different rules apply to the way you can subsequently perform admin tasks:

- **With sudo**
  If you installed from the live system or did not specify a `root` password during a standard installation, `sudo` was automatically set up. To execute admin commands, you need to switch into the admin mode in a terminal by using `sudo -s` and specifying your password (see also Chapter 10, Section 10.3).
- **Without sudo**
  Otherwise, if you performed a traditional installation and did specify a `root` password, the user created during the installation is not a member of the `sudo` group. To perform admin tasks, you need to run `su -l` in a terminal window and specify the `root` password.

SSH server

The live system installer does not install an SSH server, and the traditional installer does so only if you have enabled the appropriate option and have not overlooked it. However, if necessary, you can quickly install the SSH server:

```
root# apt update
root# apt full-upgrade
root# apt install openssh-server -y
```

Please Insert the Medium with the Name "Debian GNU/Linux"!

If you installed to a virtual machine, the default installer left a line in `/etc/apt/sources.list` that points to the virtual DVD as the installation media. Now, every time you want to install a package (`apt install`), `apt` wants you to reinsert the installation media. This doesn't make much sense as you want to download the latest version of the package from the internet.

Here is a solution to this: run `sudo gnome-text-editor /etc/apt/sources.list` and remove the line that starts with `deb cdrom`.

Enabling contrib and nonfree package sources

Several official package sources exist for Debian. By default, `main` and `non-free firmware` are active. For the vast majority of packages, this is sufficient.

However, if you also want to install packages whose source code is not freely available or which are based on nonfree libraries, you must add `contrib non-free` to the end of each line in the `/etc/apt/sources.list` file. The file should look like the following example. In the following listing, a `deb` line has been spread over two lines using \ to save space:

```
# in /etc/apt/sources.list
deb http://deb.debian.org/debian bookworm   main non-free-firmware
contrib non-free

deb http://security.debian.org/debian-security/ bookworm-security
main non-free-firmware contrib non-free

deb http://deb.debian.org/debian bookworm-updates  main non-free-
firmware contrib non-free
```

I have refrained from printing the `deb-src` lines here, which only concern the source code packages and are usually not needed.

MP3 and multimedia

Debian can already play MP3 files and the most common audio and video formats after a basic installation. Unofficial packages with additional codecs, multimedia libraries, and programs can be found in Debian-independent package sources—for example, at *https://deb-multimedia.org*.

NVIDIA graphics drivers

The binary drivers for NVIDIA graphics cards are also available as `non-free` packages. Prior to the installation, you must add the `contrib` and `non-free` package sources mentioned earlier to `/etc/apt/sources.list`:

```
root# apt install nvidia-driver
```

For more tips on installing NVIDIA drivers, see Chapter 17, Section 17.3, and the following web page: *https://wiki.debian.org/NvidiaGraphicsDrivers*.

3.2 Fedora

Fedora is a variant of RHEL. Fedora development is supported by Red Hat in terms of personnel and funding. Unlike RHEL, both Fedora itself and all updates are available free of charge. For Red Hat, Fedora is a kind of testing platform to develop and test new features. For many Linux freaks, on the other hand, Fedora is the most modern Linux distribution available. New Linux concepts and ideas are often first found in Fedora before other distributions follow suit. Usually, Fedora is also the Linux distribution that lets you try out the latest GNOME version first.

Despite the development team's willingness to experiment, Fedora has proven to be a relatively stable distribution in recent years. This is obviously where the expertise of the Red Hat developers comes into play. In terms of usability, Fedora has made great progress in recent years: while Fedora used to have the reputation of being "by freaks, for freaks," the distribution is now almost as easy to use as Ubuntu.

The biggest disadvantage of Fedora is its short lifespan: Fedora updates are maintained for the cycle of two releases plus one month. For example, the update period for Fedora 38 ends one month after Fedora 40 is completed. Because the release cycle is normally six months, this corresponds to an update period of only 13 months.

Variants

Fedora is basically available in the three stable variants, *Workstation*, *Server*, and *IoT*. Two more variants are in development:

- *CoreOS* (*https://getfedora.org/en/coreos*) is optimized for use in container environments.
- *Silverblue* (*https://silverblue.fedoraproject.org*) doesn't use the traditional package system; it uses `rpm-ostree` instead. This new package system allows atomic updates as well as undo functions for updates and package installations. It is still open for debate whether Silverblue is the future of Linux or a dead end.

Spins

I'll focus on the workstation variant here, which normally uses GNOME as the desktop. If you prefer another desktop system, you can switch to Fedora Spins. These are Fedora variants with a predefined package selection for alternative desktop systems like KDE, Xfce, LXQt, MATE, Cinnamon, LXDE, or i3 (see *https://spins.fedoraproject.org*).

CPU architectures

Furthermore, there are Fedora variants for other CPU architectures (ARM, Power, and s390x) as well as images prepared for various virtualization and cloud systems (see *https://alt.fedoraproject.org*).

In the near future, there should be a variant of Fedora for Apple computers with the M1, M2, and so on CPUs.

Documentation

For more information about Fedora, you should visit the following websites:

- *https://fedoraproject.org*
- *https://fedoraforum.org*
- *https://docs.fedoraproject.org*
- *https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux*

3.2.1 Installing Fedora

Installation media

At *https://getfedora.org*, you first choose **Fedora Workstation** and then download Fedora Media Writer for Windows, macOS, or Linux. This small program then downloads Fedora and transfers the image to a USB flash drive. Alternatively, you also can find direct download links for the live ISO image at *https://getfedora.org*, which you can then write to a USB flash drive yourself or use as installation media for a virtual machine.

When booting from the installation media, it is scanned for errors by default. You can avoid this time-consuming test by selecting **Start Fedora Workstation** using the cursor keys.

If you install Fedora on a computer with a new NVIDIA graphics card, the installation may get stuck when the graphics mode is activated. Usually, the installation succeeds if you select **Troubleshooting • Start Fedora in Basic Graphics Mode** from the boot menu. You can install the NVIDIA drivers manually at a later time.

In the live system, you can try out Fedora or start the installation program. After setting the language, this combines all configuration settings in a single dialog (see Figure 3.4). Usually, you only need to change a single item—namely, **Installation Destination**.

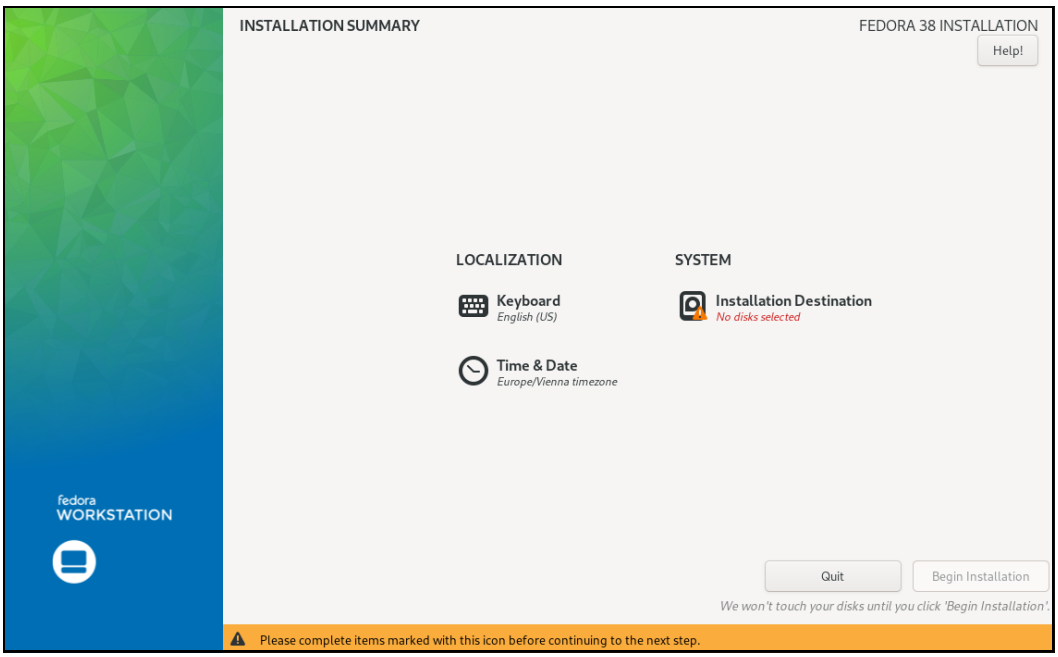


Figure 3.4 Overview of Installation Settings

Partitioning

Clicking the **Installation Destination** icon takes you to a partition editor, which unfortunately is not a masterpiece in terms of intuitive operation. To put it more drastically: I have worked with many partition editors over the last 30 years. The one provided by Fedora/RHEL is the one whose operation is the most illogical.

The first dialog lists the local hard disks and SSDs found (see Figure 3.5). You must make sure that those hard disks on which partitions are to be created or used are marked with a checkmark. (If there is only one disk, then it is already selected, and you only need to click **Done**. The checkmark is decisive. Whether or not the disk has also just been selected with the mouse and is displayed with a blue background, on the other hand, is irrelevant.)

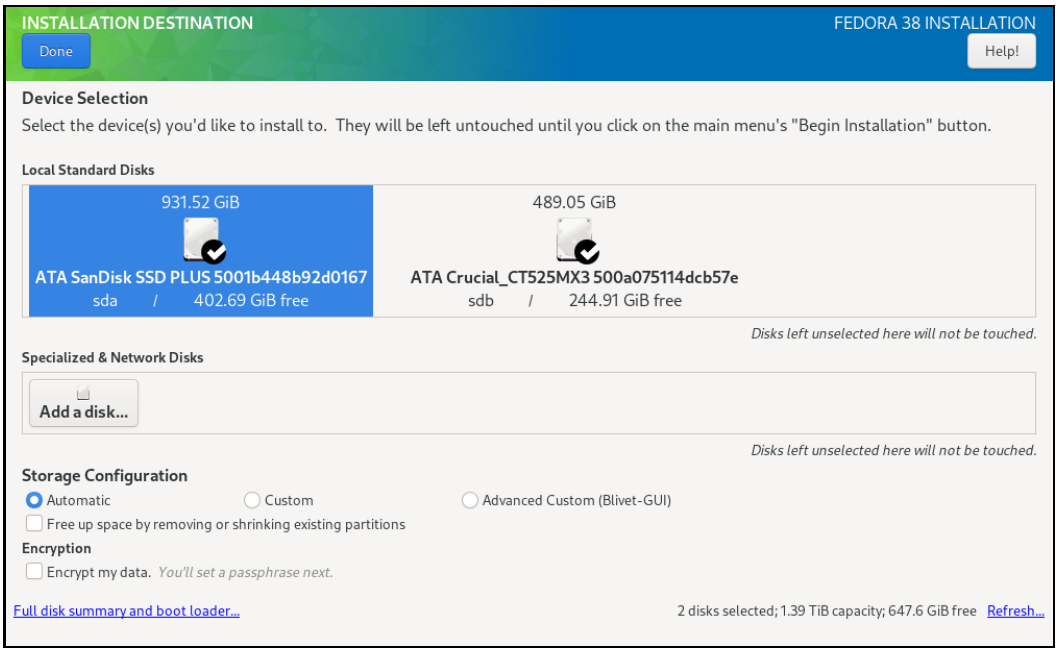


Figure 3.5 Selecting Partitioning Method

Some options enable you to control how you want to proceed:

- If you select **Automatic**, the installation program will take care of suitable partitioning. This works well when you want to make full use of a disk that is still empty—for example, when installing in a virtual machine.
- **Custom** opens an editor in the next step where you can set up the partitions manually and change (or even shrink) existing partitions.
- The **Advanced Custom (Blivet-GUI)** option also takes you to a partition editor designed for advanced users. I personally find this editor easier to

use than the default editor, but possibly that is because I am actually an advanced Linux user. :-)

- The **Free Up Space by Removing or Shrinking Existing Partitions** option is only available in combination with the **Automatic** option. It allows you to delete or shrink some partitions in advance.
- The **Encrypt My Data** option causes the LVM system to be encrypted.

Automatic partitioning

Provided that you have chosen automatic configuration and the hard disk has been empty so far, clicking **Done** actually finishes the configuration and takes you back to the main dialog. Fedora performs the partitioning on its own, but it doesn't find it worth the trouble to tell you the results of its work. Accordingly, this is my task: The installer sets up a 200 to 600 MiB EFI partition, a 1 GiB boot partition, and a third partition that fills the rest of the disk. A file system formatted using `btrfs` ends up in it. For background information on `btrfs` and its Fedora-specific configuration, see Chapter 18, Section 18.11.

Manual partitioning (default program)

There is no way to influence the automatic configuration or to modify it afterward. If you want to perform the partitioning yourself, you must select the **Custom** option. **Done** then leads to another dialog (see Figure 3.6).

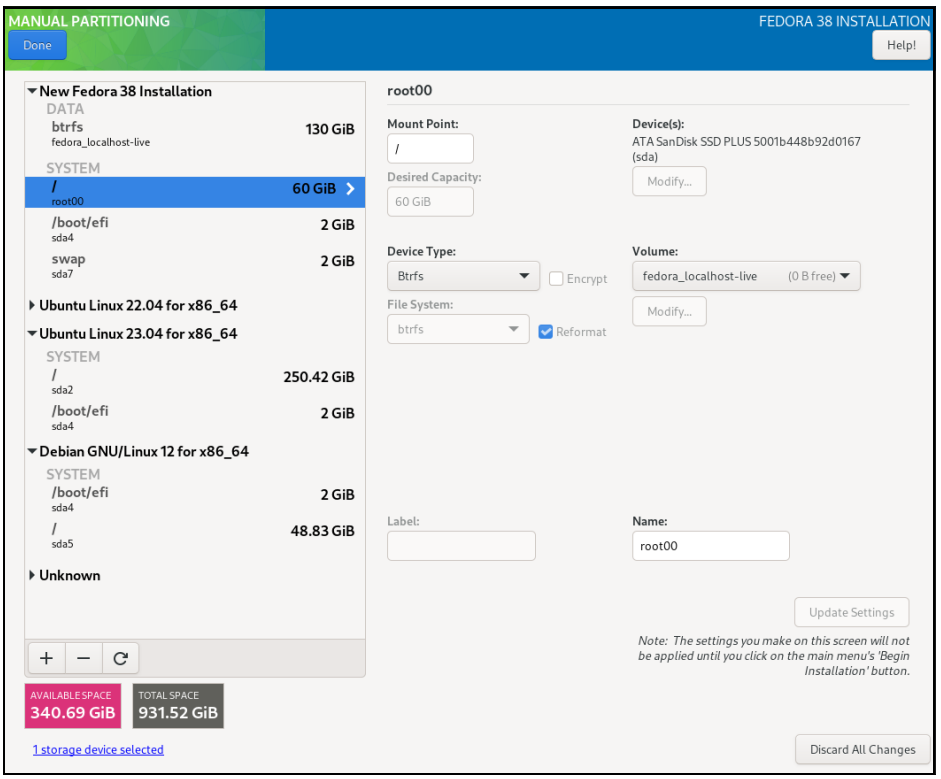


Figure 3.6 Manual Partitioning in Default Editor

The bar on the left lists all partitions or file systems that exist on the disks and those that are to be newly created. The partitions are grouped as follows:

- The first group, which is initially empty, describes the new Fedora system.
- The other groups assign the existing partitions to the already installed operating systems. It may well happen that a partition is displayed in several groups, such as a swap partition that is used by several Linux distributions in parallel.

Figure 3.6 shows an installation on a test machine in parallel with some already existing distributions. A new root partition of 60 GiB and with a `btrfs` file system is supposed to be set up for Fedora. The existing EFI partition, `/dev/sda4`, is to continue to be used under the `/boot/efi` directory. In addition, a new swap partition of 2 GiB is to be created. The remaining existing partitions are not to be addressed under Fedora for the time being.

To create space, you can delete or shrink existing partitions. To set up new partitions, click the plus button (**+**) and specify only two parameters for the time being: under **Mount Point**, enter the mount directory or the name "swap," as well as the required size in MiB or GiB.

Only in the second step do you select the file system type. Under **Device Type**, you need to specify whether the file system is to be set up in an ordinary partition or in a logical volume.

When you have set up all the partitions as you wish, you can finish the process by clicking the **Done** button in the upper-left-hand corner. The installation program then displays a summary of the pending actions. Confirming this data will take you back to the main dialog.

The Hard Disk Will Not Be Changed Until You Have Confirmed

Although the handling of the installation program takes some getting used to, there is at least one advantage: changes you make in the editor are not executed immediately. Rather, the installer waits until you are done with your configuration. Only after a confirmation prompt, which also displays a summary of all outstanding actions, will these be executed.

Manual partitioning (Blivet GUI)

The **Advanced Custom (Blivet-GUI)** and **Done** options take you to an alternative partition editor that I find more intuitive to use. It displays the hard disks, SSDs, and LVM systems found on the computer in the left-hand sidebar (see Figure 3.7).

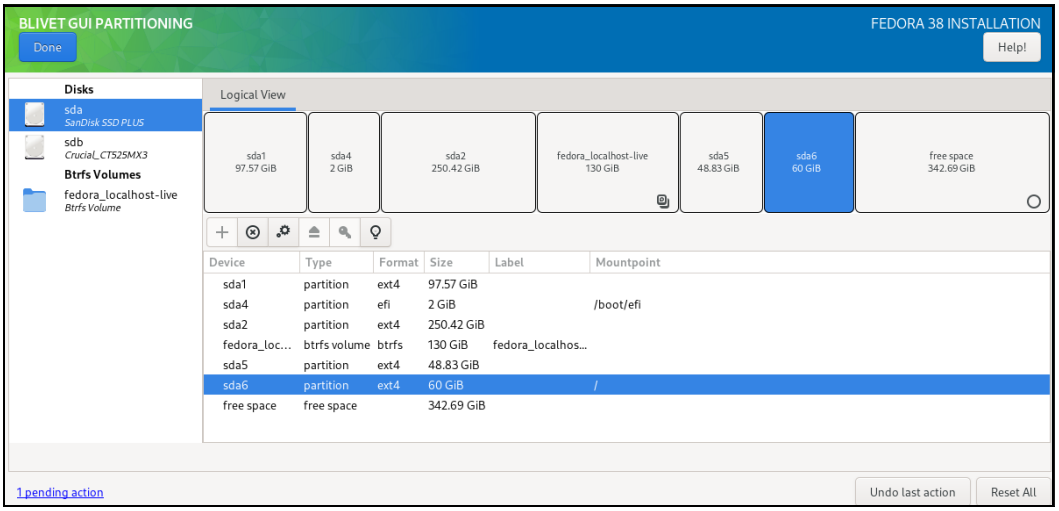


Figure 3.7 Manual Partitioning in Blivet GUI

For the currently selected element, the main area of the editor then displays the list of partitions or logical volumes (LVs). Using buttons and context menu commands, you can then change, set up new, or delete partitions and LVs.

Time zone

The installation program selects a time zone that matches the language setting. If you do not agree with this, you can use **Time & Date** to select another time zone and set the parameters of the NTP server. As soon as you finish the main dialog with **Start Installation**, the actual installation begins.

Configuring users

You must now wait for the actual installation to finish. You only have the option of setting up a user after the first restart. This user will automatically get admin rights and will thus be able to switch to `root` mode via `sudo` (see Chapter 10, Section 10.3). As with macOS and Ubuntu, the `root` user does not receive a password by default and remains locked.

3.2.2 Getting Started

First update

After the first login, you should perform an update—either using the **Software Update** program option or by entering "dnf update" in the terminal. If you choose the first option, the updates will be downloaded first. For their actual installation, the computer will then be restarted.

The first update often takes a similar amount of time as the installation. In this respect, `dnf update` is preferable because then the update is executed during operation. Although some updates (e.g., of the kernel) will only take effect with a reboot in this case, at least you can do other work in parallel.

Enabling the SSH server

By default, an SSH server gets installed, but it does not start automatically. The following command provides an uncomplicated solution to this:

```
root# systemctl enable --now sshd
```

alias settings for root

If you work as `root`, security prompts constantly appear when you run `mv` and `rm` to ask if you really want to perform the operation. These security prompts will stop when you remove the `alias` statements from `/root/.bashrc`.

Setting up additional package sources

In the official Fedora packages, some packages are missing for licensing and patent reasons: drivers for NVIDIA graphics cards, various audio and video codecs, and so on. Fortunately, these packages are available in alternative package sources, the most popular of which is *RPM Fusion* (see *https://rpmfusion.org*).

Setting up those package sources is quite simple: the **Package Sources** menu item of the software program opens a dialog in which you can activate some predefined repositories with one click (see Figure 3.8).

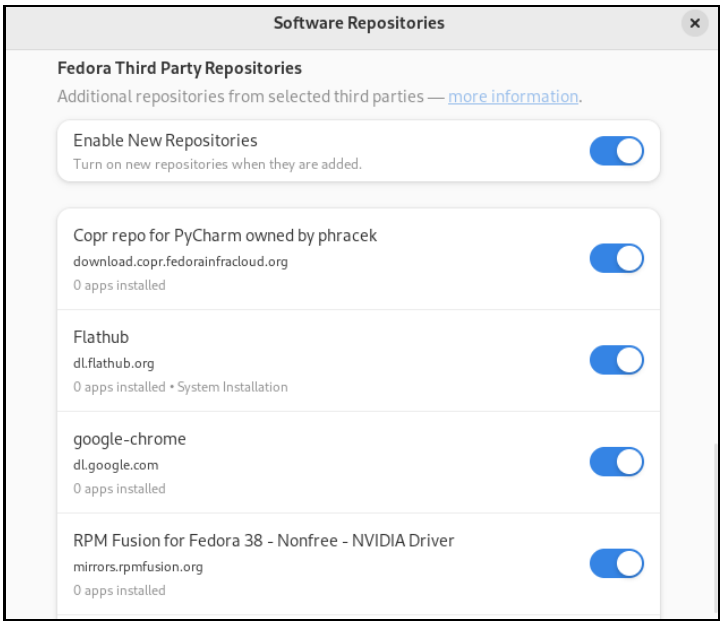


Figure 3.8 Enabling Alternative Package Sources

Multimedia packages

To install various multimedia codecs that are missing in Fedora, you first need to enable the RPM Fusion package sources and then run the following command:

```
user$ sudo dnf install gstreamer*-plugins-bad* gstreamer*-plugins-ugly*
```

NVIDIA drivers

If you depend on the binary graphics drivers from NVIDIA, there are various installation options. As a matter of fact, it should be sufficient to activate the NVIDIA package source in Software and then search for and install the relevant packages in the software as well (shown in Figure 17.3 in Chapter 17). This sometimes worked in my tests, but not always. If necessary, the following command can be executed in the terminal after activating the NVIDIA package source:

```
user$ sudo dnf update -y
user$ sudo dnf install xorg-x11-drv-nvidia akmod-nvidia
```

During the installation, `/etc/X11/xorg.conf` is automatically modified so that the new driver is automatically used after a reboot of the computer. Two alternative ways to install NVIDIA drivers are described in the following web pages:

- *https://negativo17.org/nvidia-driver*
- *https://if-not-true-then-false.com/2015/fedora-nvidia-guide*

Don't be fooled by the year 2015 in the second link! This is when the article was first published. Since then, it has been updated countless times for the current Fedora version.

3.3 Linux Mint

Linux Mint (*https://linuxmint.com*) is a very popular variant for Ubuntu Linux. Linux Mint is based on the same package sources as Ubuntu, generally using only LTS versions of Ubuntu. The Linux Mint version 21.2 tested here is based on Ubuntu 22.04. Instead of the original GNOME desktop, however, Mint uses the MATE or Cinnamon desktop system, depending on the variant.

Linux Mint differs from Ubuntu in other aspects as well:

- Mint uses its own file manager, its own system settings modules, and various smaller add-on programs, and thus it also differentiates itself functionally from Ubuntu. The goal of all in-house developments is simpler operation.
- In addition to the Ubuntu package sources, Mint uses its own package sources. Not only do these serve to provide packages not provided for in Ubuntu, but they also allow Mint to replace individual Ubuntu packages with newer versions.

  Basically, this is a good thing. However, Mint's own package source may prove to be a disadvantage if you set up other package sources (personal

package archives [PPAs], package sources for Docker, etc.) to install external add-on packages. Incompatibilities occur time and again in the process. Especially for software developers, it can therefore be advantageous to stay with the original (i.e., Ubuntu).

- Mint avoids the Snap package format favored by Canonical. For the Firefox web browser, which is now only available as a Snap package on Ubuntu, Mint itself maintains a traditional DEB package.
- The user interface of Mint can be designed much more extensively than that of Ubuntu. Besides icons, miniprograms (called *desklets*) can also be placed on the desktop.

Unlike Ubuntu, Lubuntu, and so on, Linux Mint is not one of the "official" Ubuntu derivatives.

Variants

Linux Mint is available in various flavors, which differ in the desktop system used. (You can choose the proprietary Cinnamon development or MATE or Xfce). In this section, I will only discuss the most popular variant: Linux Mint with the Cinnamon desktop.

Installation

ISO media for Linux Mint is available for download at *https://www.linuxmint.com/download.php*. As with the Debian live system (see Section 3.1), the installation program is based on Calamares and is extremely easy to use.

EFI

Like Ubuntu, Linux Mint uses the `/boot/efi/EFI/ubuntu` directory for installing the boot loader files. This causes one distribution to overwrite the boot files of the other distribution when Ubuntu and Linux Mint are installed in parallel.

Cinnamon

The Cinnamon desktop is basically based on GNOME 3 and its libraries. Instead of `gnome-shell`, Mint runs its own window manager Muffin, and instead of the Nautilus file manager, it runs its fork, Nemo. Visually, Cinnamon looks more like GNOME 2 than GNOME 3, and that is quite intentional. Figure 3.9 shows the Cinnamon desktop of Linux Mint with the characteristic start menu and the welcome wizard, which helps with the installation of programs, proprietary drivers, and multimedia codecs, among other things.

Package management

Package management works like in Ubuntu (see Chapter 16, Section 16.6). However, the `/etc/apt/sources.list.d/official-package-repositories.list` file contains another Mint-specific package source in addition to the Ubuntu LTS package source. At the command level, you can use `apt` to update the entire system as well as install individual packages.

In contrast to Ubuntu, the Snap package system is not available by default, whereas flatpaks are active (see also Chapter 16, Section 16.11).

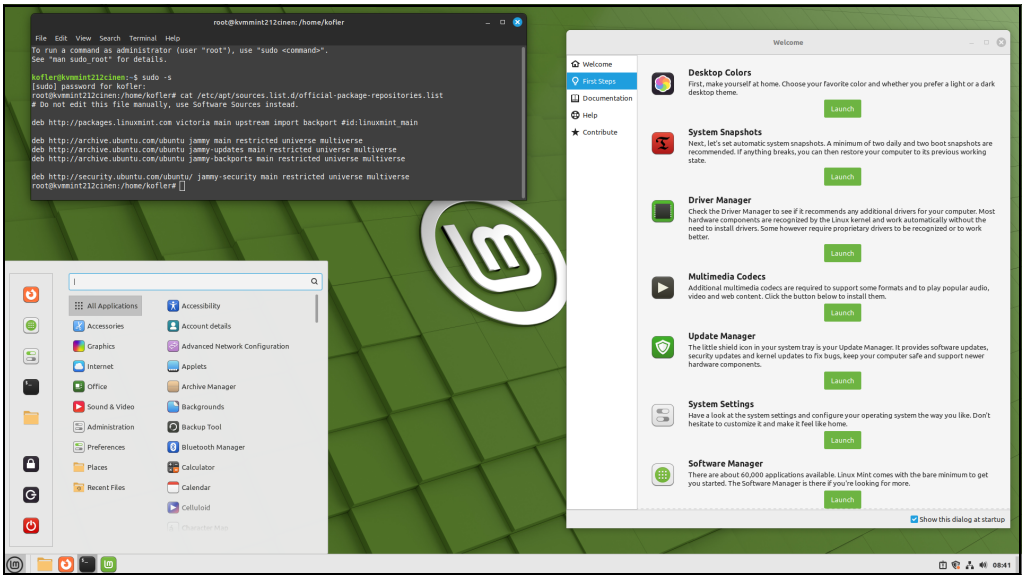


Figure 3.9 Cinnamon Desktop in Linux Mint

The `mintupdate` program takes care of installing updates, and you must confirm each update beforehand. Automation of updates is possible, but the program recommends enabling the Mint-specific *system snapshots* feature upfront: the associated `timeshift` program creates regular backups of the operating system via `rsync`. This function makes it possible to restore a partially destroyed system (e.g., after a package installation) to an older state. However, the snapshot function does not include personal data, so it is not intended for backups.

Mint-specific programs

Besides `mintupdate` and `timeshift`, there are a number of other Mint-specific programs. The easiest way to find their names is to type "mint" in a terminal window and press Tab. The following list names the most important representatives:

- `mintbackup`
  This is a simple backup tool.
- `mintdrivers`
  This supports you with the installation of hardware drivers similar to Ubuntu.
- `mintinstall`
  This helps to install programs. This Mint-specific variant of GNOME's Software takes into account both the Ubuntu and Mint package sources as well as flatpaks.
- `mintstick`
  This helps to write an ISO file to a USB flash drive.

3.4 Manjaro Linux

Arch Linux

Before I can explain the benefits of Manjaro, I must briefly discuss its basic distribution. Arch Linux, which has been available for 20 years, sees itself as a modern rolling release distribution for advanced Linux users. The installation is done in text mode, and a lot of manual work is involved in the configuration. Even Linux professionals still learn a lot in the process. Arch Linux uses its own package management system apart from the established RPM and DEB paths (see Chapter 16, Section 16.7).

Even though I personally find Arch Linux very attractive and have been using it on my most important notebook computer for over two years, the distribution is unsuitable for Linux beginners and is therefore out of place in this chapter. The right time to switch to Arch Linux from another Linux distribution comes after reading Part V of this book. If you have already acquired Linux know-how with other distributions and would like to make friends with Arch Linux, there are many installation guides on the internet (see, for example, *https://wiki.archlinux.org/title/installation_guide*).

Manjaro Linux

There are some distributions derived from Arch Linux. The best known is Manjaro Linux, hereafter *Manjaro* for short. The two main differences compared to Arch Linux are the addition of an easy-to-use installer that is launched from a live system and the integration of a convenient package management and software update program. Nevertheless, most of the advantages of Arch Linux remain. You can download the distribution from *https://manjaro.org*.

The biggest disadvantage of Manjaro is its own package sources. While they protect Manjaro users from mischief caused by updates made too early to Arch packages, they also make manual intervention more difficult. If you need specific additional packages or if problems with the package management occur, you have an advantage with the original (i.e., Arch Linux). The great documentation for Arch Linux doesn't help because it doesn't deal with Manjaro-specific peculiarities.

No UEFI Secure Boot

Neither Manjaro nor the underlying Arch Linux support UEFI Secure Boot. You may need to disable this security feature before starting the installation. Manjaro uses GRUB as a boot loader.

Installation

The installation is performed from within a live system. Depending on which desktop system you want to use, there are several live images for GNOME, KDE, and Xfce to choose from on Manjaro's download page. I tested the GNOME variant for this chapter.

The Manjaro development team already shows its attention to detail in the boot menu of the installation system, where you can set the language,

keyboard layout, and time zone (see Figure 3.10). With most other distributions, the first installation dialogs are always displayed in English until the language setting is done. You also have the option to run the installation system with proprietary drivers, which is especially helpful on computers with modern graphics cards.

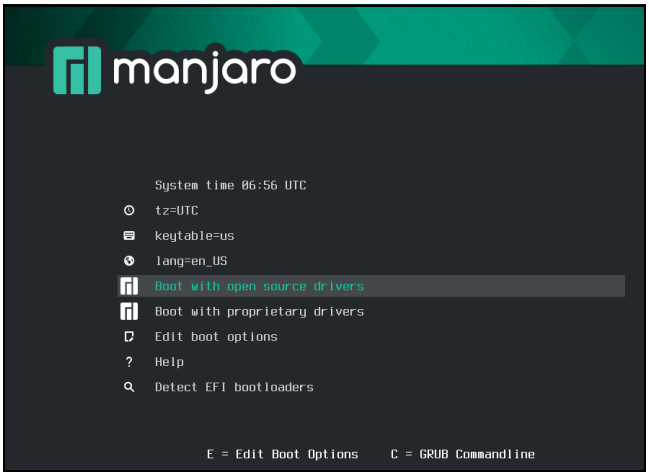


Figure 3.10 Language and Time Zone Settings in Boot Menu

In the desktop of the live system, you then start the graphical installation program. Language and keyboard layout are taken from the boot settings, but you have to select the time zone again. You can perform the partitioning either automatically or manually (see Figure 3.11). Unfortunately, because the Manjaro developers love dark colors, the dialogs are quite difficult to read.

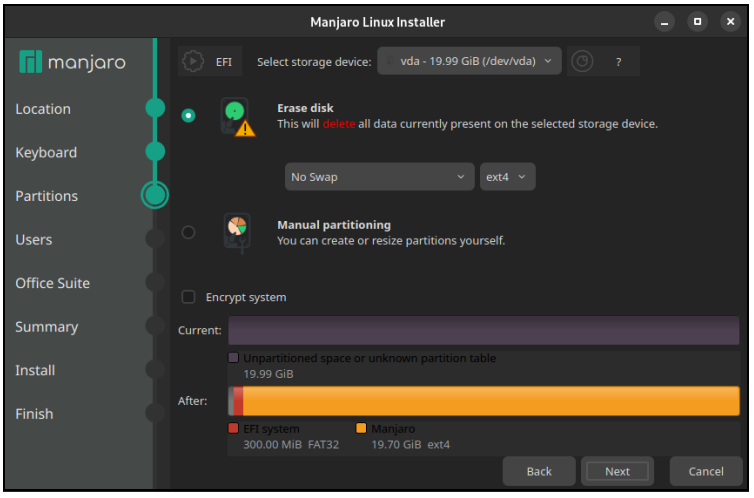


Figure 3.11 Automatic Partitioning of Data Medium

With automatic partitioning, you have the choice of setting up your system without swap space, with a swap partition, or with a swap file, and you choose whether or not you want to encrypt the system partition.

zsh instead of bash

Manjaro is the only one of the distributions presented in this chapter that activates `zsh` by default instead of the otherwise widely used `bash`. Along the way, various shell extensions are also set up; the terminal leaves a somewhat playful impression (as suggested by Figure 8.1 in Chapter 8). The configuration largely corresponds to my personal shell preferences. If you prefer to stick with traditional `bash`, you can simply run `chsh -s /bin/bash` in the terminal and then log in again.

Package management

Like Arch Linux, Manjaro uses the `pacman` command for package management (see also Chapter 16, Section 16.7). Alternatively, however, the *Pamac* graphical software management tool is installed (see Figure 3.12) to help you install and update packages. The website at *https://software.manjaro.org/applications* contains a collection of popular programs that can be installed with the click of a button via Pamac.

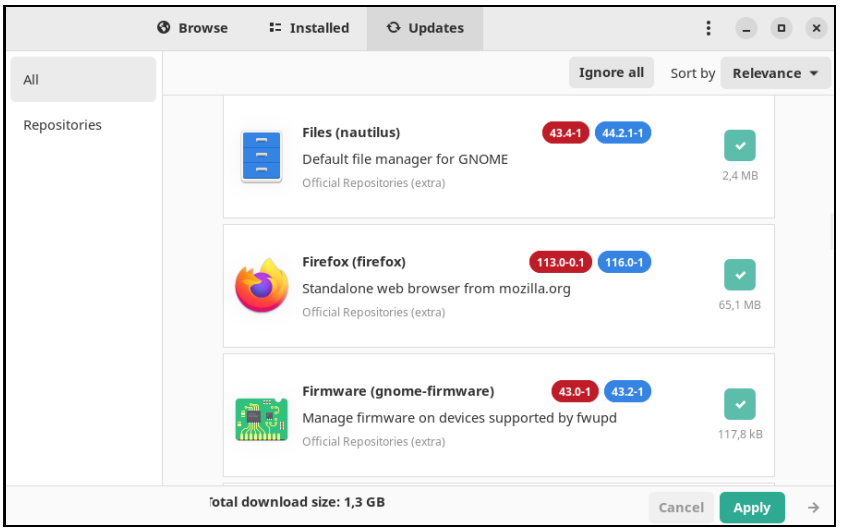


Figure 3.12 Pamac Package Manager

Manjaro uses the same package format as Arch Linux but its own package sources. Arch Linux security updates are applied immediately, other packages often a bit later. The logic as to which packages from Arch Linux are adopted into Manjaro and when is outlined at *https://manjaro.org/features/fresh-and-stable*.

3.5 openSUSE

openSUSE Leap (hereafter mostly shortened to *openSUSE*) is a widely used Linux distribution, especially in German-speaking countries. A key differentiator of the various SUSE distributions compared to their competition is the all-encompassing configuration and administration tool YaST (for *Yet another Setup Tool*).

openSUSE is also considered one of the best KDE distributions (see Figure 3.13). openSUSE thus stands in contrast to most other distributions that primarily focus on GNOME. (Note, however, that the enterprise variants of SUSE are also based on GNOME.)

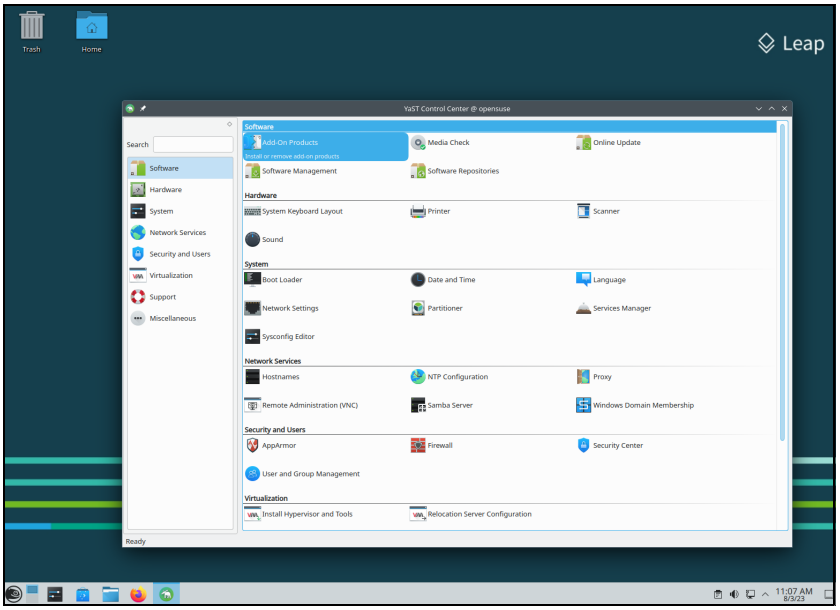


Figure 3.13 KDE Desktop of openSUSE with YaST Configuration Program

The name SUSE

SUSE (then spelled SuSE) was originally a German company, and the abbreviation SUSE originally was derived from *Gesellschaft für Software und Systementwicklung* (Society for Software and System Development). In 2003, Novell acquired SUSE. After that, SUSE changed hands several times before becoming a publicly traded company in 2021.

openSUSE Leap

openSUSE Leap is a free variant of the commercial SUSE distributions. The development is heavily driven by SUSE staff, but there are public beta releases, mailing lists, a bug database, and an active community that participates in and supports development.

SUSE Linux Enterprise Server (SLES) in the same version number serves as the substructure for openSUSE. The kernel, the graphics system, and other

basic components are thus relatively conservatively specified. The openSUSE development team is focused on providing openSUSE with up-to-date versions of programming languages and desktop components (see *https://www.opensuse.org* and *https://doc.opensuse.org*).

Living in the past

In August 2023, when I wrote this chapter, openSUSE 15.5 was the latest version. However, since it is also based on SUSE Linux Enterprise 15 from 2018, some core components are hopelessly outdated. For example, it ships with Python version 3.6, although version 3.11 would be current at that time; `bash` is version 4.4 (currently 5.1); GNOME is version 41 (currently 44); and so on. In this respect, openSUSE 15.*n* appears to be only of limited value at the moment. The rolling release variant Tumbleweed, which I will discuss ahead, is more exciting.

Unclear future

For the next version of Enterprise Linux, SUSE is working on a complete rebuild. SLES 15 is to be replaced by the *Adaptable Linux Platform* (ALP). ALP is based on MicroOS, a new type of *immutable* distribution in which updates take the shape of atomic transactions, not package updates as in the past. However, SUSE ALP is only the core operating system. The applications installed in it should run similarly to containers (think of Docker or Podman).

It is still completely open what this means for the future of openSUSE. First, openSUSE 15.6 is scheduled to be released in 2024 as the last version in the current cycle. To create a new openSUSE version based on ALP, openSUSE is looking for community support. However, because ALP is designed as a pure server platform without desktop packages, this could prove difficult.

Tumbleweed

One very interesting alternative to openSUSE Leap is Tumbleweed. This is an openSUSE variant designed as a *rolling release*: once installed, this distribution constantly receives more current software versions as part of the update system so that (at least in theory) a new installation or a distribution update will never be necessary again.

The Tumbleweed project is based on *Factory*, the development branch of openSUSE. The Tumbleweed developers do make an effort to release new software versions only when the programs run reasonably stably. Nevertheless, when using Tumbleweed, problems are of course to be expected occasionally, as a new software version may still contain bugs or cause incompatibilities with other components.

My experience with Tumbleweed has been quite positive. Tumbleweed contains much more recent software versions (in August 2023: kernel 6.4, bash 5.1, Python 3.11, etc.) than openSUSE Leap.

Note that Tumbleweed does not provide updates to existing packages, but simply new versions of the packages. For this reason, you should use `zypper`

`dup` instead of `zypper up` to update the distribution. For more Tumbleweed tips, visit *https://en.opensuse.org/Portal:Tumbleweed*.

3.5.1 Installing openSUSE

The following installation instructions are for openSUSE Leap 15.5. Basically, however, the instructions also apply to Tumbleweed, whose installation is quite similar. At *https://get.opensuse.org/desktop*, you can find download links for both openSUSE variants and various hardware platforms. You also have a choice between the image with the default installer (the offline image, approximately 4 GiB) and live variants for selected desktop systems. I always use the offline image, which provides more options to influence the installation process.

Starting the installation

In the first dialog of the installation program, you set the language and keyboard layout. On the page that follows, you can add additional package sources and finally specify which function you want your computer to perform. The choices include **Desktop with KDE Plasma**, **Desktop with GNOME**, and **Server**. I have decided to use the KDE desktop here.

Partitioning

The installation program then makes a suggestion for partitioning the hard disk (see Figure 3.14). By default, the program sets up an EFI partition, a swap partition, and a large system partition with the `btrfs` file system. Within the `btrfs` file system, openSUSE creates countless subvolumes.

If you agree with the proposal, just click **Next**. Otherwise, there are three other options:

- **Guided Setup** first opens a dialog where you can enable LVM and disk encryption. In the subsequent dialog, you need to specify whether you want a separate home partition and which file system type you want to use for the root and home file systems.

  The installation program suggests the `btrfs` file system for the root partition, with the **Enable Snapshots** option by default. In combination with the Snapper library, you can then undo admin tasks later. In itself, this is a fine thing. However, the snapshots require a lot of space on the hard disk/SSD and add a lot of complexity that even gives Linux professionals a hard time. You can read the details in Chapter 18, Section 18.11.

  The **Propose Separate Swap Volume** option decides whether a separate partition or logical volume should be provided for the swap space. If you disable this option, a swap file will be set up instead.

- **Expert Partitioner • Start with Current Proposal** takes over the current partitioning proposal. You can then resize the partitions, add more partitions, and so on.

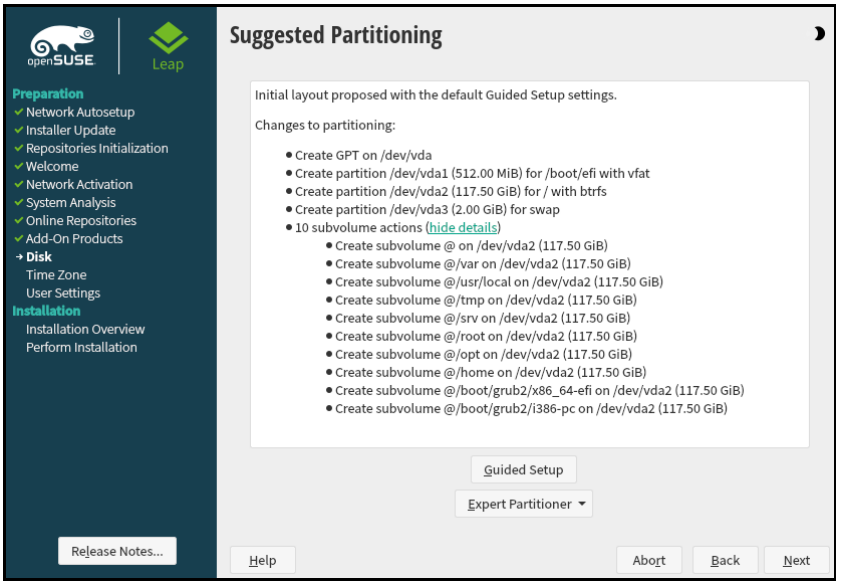


Figure 3.14 Partitioning Suggestion from Installation Program

- **Expert Partitioner • Start with Existing Partitions** leads to the same editor, but now shows only the partitions that already exist on the disks. You can now set up RAID and LVM, create partitions or logical volumes, and so on, to your heart's content (see Figure 3.15).

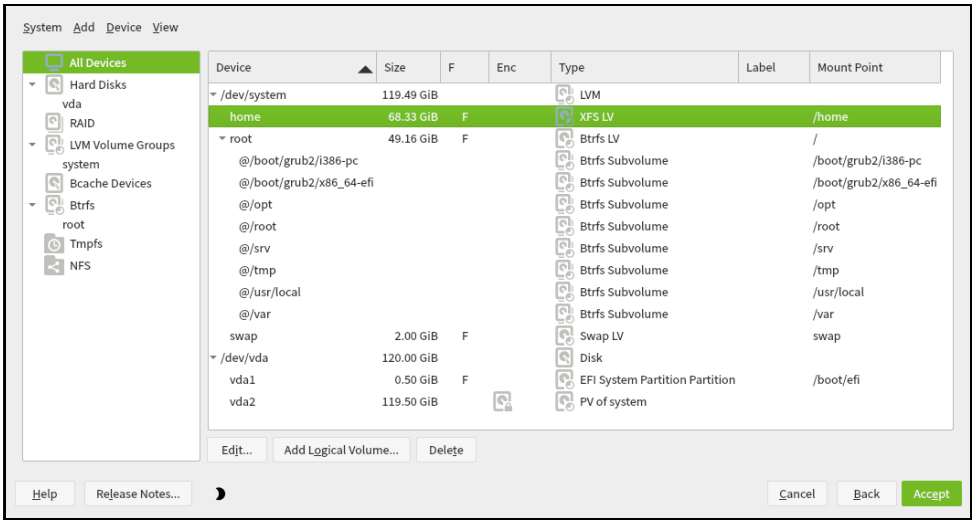


Figure 3.15 Installer's Great Partition Editor

In all three variants, **Back** takes you back to the main dialog where you can choose another option if necessary.

Praise and Criticism

In the SUSE/openSUSE installation program, even complex setups for organizing the file system can be set up quickly and easily. This installation point is organized in a way more user-friendly manner than all other distributions I know. That's great!

However, I don't think it's a good idea to default to `btrfs` with countless subvolumes as the file system type. If you are not (yet) a Linux expert, I strongly advise you to do a **Guided Setup** and set the `ext4` file system type for the root partition!

Basic settings After partitioning, you want to set the time zone and set up a new user. Rather unusual is the fact that the user password is also valid for `root` by default. Only if you disable the related option can you specify your own `root` password in the dialog that follows. The SUSE developers justify their approach with the fact that more than 75% of all users use the same password for `root` as for the first user account anyway. That may be, but from a safety point of view, it is still not ideal!

Summary The installation program then displays a summary of all settings (see Figure 3.16).

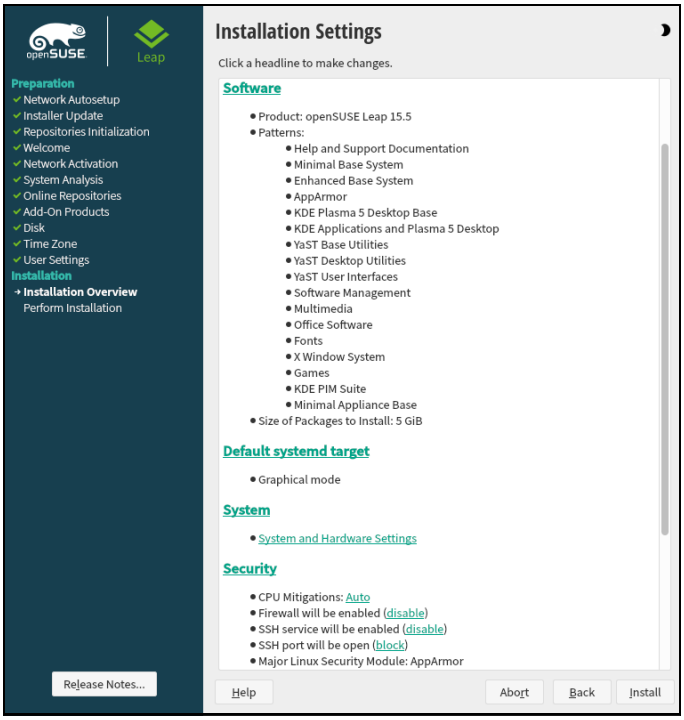


Figure 3.16 Summary of Installation Settings

If you agree, just click **Install** and off you go. However, you should take the trouble to read the installation suggestion at your leisure beforehand! It is often useful or necessary to change details. To make a change, simply click the appropriate item in the summary.

Performing the installation

Once you agree with all the settings, click the **Install** button. The installation takes a few minutes. After that, the computer will be restarted.

3.5.2 Getting Started

Setting the hostname

During the installation, you have no option to set the hostname. openSUSE uses a random string like `linux-q2uf` instead. As a workaround, in the YaST configuration program, open the **System • Network Settings** module. In the **Hostname/DNS** dialog sheet, you now specify the desired hostname. In order for all KDE and GNOME programs to track this change, you must log out and log in again.

Setting the firewall

In openSUSE, a firewall is installed and also active by default. This blocks almost all outgoing network traffic. Among other things, this makes it impossible to access Windows or Samba network shares. The configuration is done by the unfortunately largely useless YaST module at **Security • Firewall**. For this reason, it's better to use the `firewall-cmd` command to set the firewall (see Chapter 29, Section 29.4).

Multimedia

Although openSUSE comes with various audio and video players out of the box, multimedia support is relatively poor as many audio and video formats cannot be played.

The solution to this problem is the Packman package source. To enable it, start the YaST **Software Repositories** module, run **Add • Community Repositories** there, and enable the Packman package source. When doing so, you must confirm the Packman package source key as trusted.

After that, start the YaST **Software Management** module, select the Packman package source via **View • Repositories**, and then click the **Switch System Packages to the Versions in This Repository** link.

NVIDIA graphics drivers

If your computer contains an NVIDIA graphics card, the `nouveau` driver is used by default. The driver works well with many models, but it cannot make optimal use of the graphics card's energy-saving features.

In previous openSUSE versions, installation of the proprietary NVIDIA driver was very easy: to do this, you can use YaST to activate the NVIDIA package source. Next, start the YaST **Online Update** module and then execute the menu command **Extras • Install All Matching Recommended Packages**. YaST then determines which GPU your NVIDIA graphics card contains and installs the appropriate driver package. (There are several different

packages, and it's not very easy to guess which package goes with which graphics card.)

However, during my tests, the activation of the NVIDIA package source failed. An internet search revealed that instead of *https://download.nvidia.com/opensuse/leap/15.5*, the address *http://download.nvidia.com/opensuse/leap/15.5* must be used (i.e., HTTP instead of HTTPS). To set up the package source, you can use **Add • Specify URL**. For more tips on using the NVIDIA drivers on openSUSE, visit *https://en.opensuse.org/SDB:NVIDIA_drivers*.

3.6 Pop!_OS

The Pop!_OS Linux distribution is derived from Ubuntu and is developed by an American company called *System76*. This company sells notebook computers with Pop!_OS installed by default. Pop!_OS can be downloaded free of charge from the following website and then, of course, installed on any computer: *https://pop.system76.com*.

Pop!_OS differs from the Ubuntu base in a surprising number of details:

- **Cosmic DE**
  Currently (i.e., in Pop!_OS 22.04), Cosmic DE is a heavily modified GNOME desktop with its own icons, theme, shortcuts, and various preinstalled extensions. In future Pop!_OS versions, Cosmic DE should become (more) independent from GNOME.
- **Boot process**
  Note that systemd-boot is used instead of GRUB (see Chapter 19, Section 19.5).
- **Package sources**
  There is a separate DEB package source with numerous own or modified packages compared to Ubuntu, as well as Flatpak instead of Snap.
- **Versions**
  After a while of doing semiannual releases, Pop!_OS is now on a two-year cycle. The current Pop!_OS version 22.04 is based on Ubuntu 22.04 LTS. The next version will probably not be available until spring 2024.

Target audience and future

According to its own definition, Pop!_OS is especially optimized for software developers. In the past, the excellent NVIDIA support was also a unique selling point. Meanwhile, setting up a computer with NVIDIA graphics is no longer witchcraft, even with most other distributions.

As a matter of fact, at the beginning I was skeptical about whether Pop!_OS could establish itself in the huge market of Linux distributions in the long

run. After all, the relatively small System76 company has to maintain a wealth of its own software and provide the distribution with bug fixes and software updates. In the past, many distributors have failed at this point. However, you need to give System76 credit for the fact that Pop!_OS maintenance has been working well since 2017 now.

Cosmic DE

But that's not all: System76 is dissatisfied with central parts of the GNOME desktop and has decided to develop its own desktop environment under the name *Cosmic DE*. However, it is unclear when this new, GNOME-independent version of Cosmic DE will actually ship. Most recently, System76 has regularly reported on various Cosmic DE components on its company blog, but there has been no test version or even a release plan available (see *https://blog.system76.com*).

No UEFI Secure Boot

Due to the possible integration of proprietary NVIDIA drivers, Pop!_OS does not support UEFI Secure Boot (even if these drivers are not used at all). You may need to disable this security feature before starting the installation.

3.6.1 Installing Pop!_OS

Pop!_OS is available on the System76 website in two variants. These differ in whether the NVIDIA driver is integrated in the image or not. Note that currently, due to licensing issues, no other major Linux distributor provides the proprietary drivers directly; rather, the drivers must be downloaded during or after installation. Pop!_OS has an advantage here in that the NVIDIA drivers are available right from the start—and this is independent of the network connection, which might still cause problems during the installation. Possibly System76, as a notebook manufacturer with good NVIDIA contacts, has a better negotiating position here.

Pop!_OS is installed from a live system, just like Fedora or Ubuntu. System76 has developed its own installation program and decorated it with its own original pictures (see Figure 3.17).

Whether you'll be happy with the installer depends on the framework:

- The installation process is extremely simple when Pop!_OS is allowed to use the entire SSD and does not have to take into account any other operating systems that may be present.
- In all other cases, and especially in the case of a parallel installation to Windows or to another Linux distribution, you must perform the partitioning manually. This gives Linux professionals great flexibility, such as when integrating existing LVM setups including encryption. However, Linux beginners will be hopelessly overwhelmed at this point.

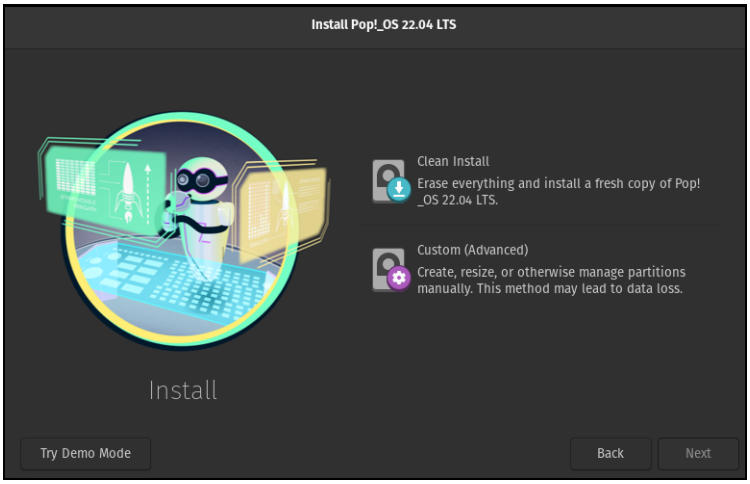


Figure 3.17 Pop!_OS Installation Options

Using the entire disk

In the first case, after selecting the **Clean Install** option and clicking the desired SSD, you are almost done: you only need to provide a name and password for your account. This password is also used by default to encrypt the file system. If you want, you can set a different password here or dispense with encryption altogether. There are no other questions or options.

In my tests, after rebooting, I had to enter the encryption password on a completely black screen that actually gave the impression that Pop!_OS had crashed. Only after the password was entered blindly did the actual boot process begin, and a few seconds later the desktop appeared.

The solution was a complete update and another reboot. After that, the display of the password dialog during the boot process worked properly.

Manual partitioning

In the second case (i.e., with the **Custom** installation variant), a dialog with all partitions of the disk appears in the next step. However, the installation program doesn't give you the option to change the partitioning. Rather, **Modify Partitions** launches the GParted program (see Chapter 18, Section 18.6). You can then set up partitions using this program.

Pop!_OS requires at least two partitions: the EFI system partition (`/boot/efi`), which must be at least 1 GiB in size and contain a fat32 file system, and a system partition or corresponding logical volume.

If you also want to use LVM or encrypt a partition, you must do this work yourself in the terminal. You will learn the commands required for this in Chapter 18.

You may also have to take care of encryption yourself and run `cryptsetup` in a terminal. However, the process assumes that you are already very familiar with Linux (see Chapter 18, Section 18.19).

Once you've set up all the partitions or logical volumes you want, you need to return to the installation program. It shows all newly created partitions or logical volumes. You can then specify how you want to use the partitions—for example, for the root file system.

The Pop!_OS installation program recognizes the newly created partitions or logical volumes. A few mouse clicks then lead you to configuration dialogs in which you can set how the configured partitions or logical volumes are to be used (see Figure 3.18).

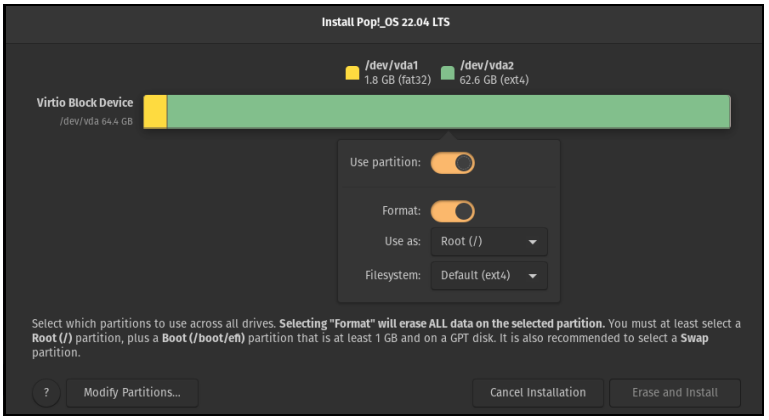


Figure 3.18 Root Partition /dev/vda2 for ext4 File System to Be Set Up

3.6.2 Getting Started

After logging in, the Cosmos desktop shows up as a GNOME system with a large number of modifications and configuration options (see Figure 3.19).

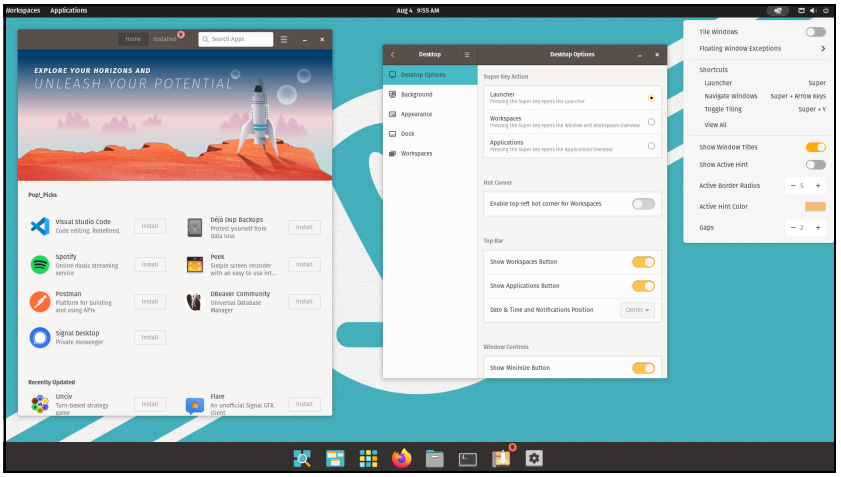


Figure 3.19 Pop Shop for Software Installation (Left); Comprehensive Configuration Options for Desktop (Middle); Menu for Optional Tiling Functions (Right)

Pop!_OS functions

The features Pop!_OS has built into the system menu of the GNOME desktop are noteworthy. On the one hand, you can choose among three different performance levels of the CPU here, and on the other hand, you can switch between Intel and NVIDIA graphics. Behind the scenes, Pop!_OS modifies the boot files for the GPU conversion. Accordingly, the changed GPU setting only takes effect after a reboot, which is a bit tedious in practice. The desired CPU profile or GPU can alternatively be set using the `system76-power` command:

```
user$ sudo system76-power graphics nvidia    (requires a reboot)
user$ sudo system76-power graphics intel     (requires a reboot)
user$ sudo system76-power profile battery
user$ sudo system76-power profile balanced
user$ sudo system76-power profile performance
```

By the way, you can enjoy `system76-power` and its GNOME extension even without Pop!_OS. To do this, set up the System76 package source on Ubuntu and install the `system76` package:

```
user$ sudo sudo apt-add-repository ppa:system76-dev/stable
user$ sudo sudo apt install gnome-shell-extension-system76-power
system76-power
user$ sudo gnome-shell-extension-prefs
```

3.7 Ubuntu

Ubuntu is currently the most popular distribution and the most widely used in the private sector. The motto of Ubuntu is *Linux for human beings*—so in a sense, it's the "human Linux." The Zulu word *ubuntu* also means *humanity toward others* or *mindful cooperation* or *I am what I am because of who we all are*. So Ubuntu is not supposed to be just a lot of software technology, but an entire philosophy.

Canonical

The company behind Ubuntu is Canonical Ltd., which is owned by the South African millionaire Mark Shuttleworth—former owner of Thawte Consulting. Compared to Red Hat, however, Canonical has far fewer employees. In the past, Canonical literally danced on all cylinders and tried to produce Ubuntu versions for smartphones, tablets, TVs, and the cloud in addition to the classic Ubuntu for PCs.

This has been over since 2017. Canonical has since focused on cloud and server customers; that's where Canonical is successful, and that's where

money can be made. The development of a standalone Linux desktop (*Unity*) was discontinued in favor of the GNOME desktop. In general, the budget for optimizations and in-house developments in the desktop area has obviously been severely limited since then.

Standard versions

There are new Ubuntu releases every six months, with the version number reflecting the date of completion. So Ubuntu 23.04 refers to the Ubuntu version completed in April 2023. Also, each Ubuntu version has a strange codename; for version 21.04, for example, it is *Hirsute Hippo*. These code names are perfect for search queries! A search for "hippo nvidia" will return much more specific results than a search for "ubuntu nvidia" or even "linux nvidia".

There is only a nine-month update service for the semiannual Ubuntu versions. Thus, these versions are actually only recommended to Linux professionals who are not bothered by regular distribution updates.

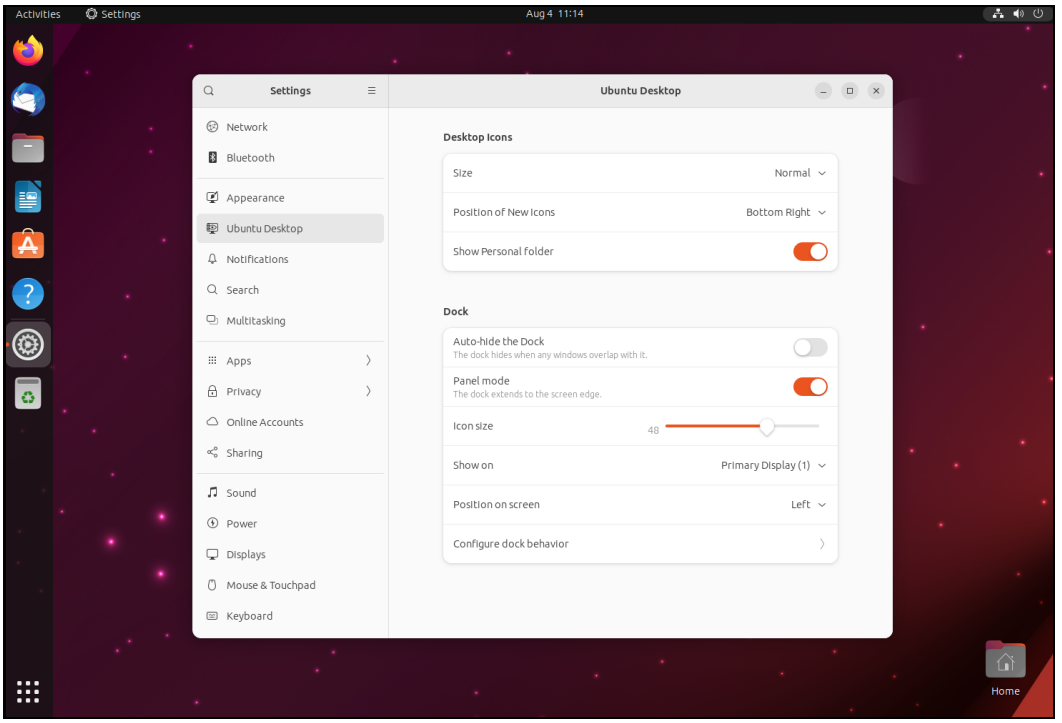


Figure 3.20 Ubuntu's Slightly Modified GNOME Desktop

LTS versions

The long-term support (LTS) versions, whose desktop packages and server packages are maintained for five years, occupy a special position among the

many Ubuntu versions. When I finished this book, 22.04 was the last available LTS version. The next LTS version, Ubuntu 24.04, will not be available until April 2024.

Variants

There are numerous distributions derived from Ubuntu. Table 3.2 summarizes the most important official—that is, supported by Canonical—ones, as well as some unofficial variants.

All official variants use the same package sources and can therefore be extended as you like. You can also install Xubuntu first and add the GNOME packages from Ubuntu later. The main difference between the different Ubuntu variants is which package selection is included on the disk and installed for the first time. The unofficial Ubuntu variants also provide additional package sources with additional packages. They contain programs that are missing from the official package sources or are included there in a different (mostly older) version.

Variant	Description
Kubuntu	Ubuntu with KDE
Ubuntu MATE	Ubuntu with the MATE desktop
Xubuntu	Ubuntu with Xfce
Lubuntu	Ubuntu with LXDE
Ubuntu Server	Ubuntu for server use (see Chapter 22, Section 22.3)
Ubuntu Studio	Ubuntu for multimedia users
Ubuntu Budgie	Ubuntu with Budgie desktop
Linux Mint	Popular Ubuntu variant without Unity (unofficial)
Pop!_OS	Ubuntu variant by System76 with especially good support for notebook computers with NVIDIA graphics (unofficial)
Elementary OS	Ubuntu variant with macOS-like desktop (unofficial)
Zorin OS	Desktop variant, which is specifically aimed at Windows migrants (unofficial)

Table 3.2 Ubuntu Variants

3.7.1 Installing Ubuntu

Installation media

At *https://www.ubuntu.com/download*, you can download free Ubuntu installation media. The ISO images must then be transferred to a USB flash drive.

Alternative Ubuntu Downloads

At *http://cdimage.ubuntu.com*, you will find various alternative installation media. This includes the daily images of the next Ubuntu version and variants currently under development.

Starting the standard installation

To perform a standard Ubuntu installation, you want to reboot the computer with the Ubuntu DVD or an appropriate USB flash drive. If you install on a computer with an NVIDIA GPU, you should select the **Safe Graphics** variant in the boot menu; otherwise, the graphics system may fail to start.

New installer

The installer described in this section has only been in use since version 23.04. In my tests, the installer worked wonderfully in virtual machines, but failed in installations on real hardware or more complex setups. By the time this book is published, there will already be the next Ubuntu version available, in which the worst teething troubles of the installer are presumably fixed.

In the live system, you can now both try out and install Ubuntu. After starting the installation program, you need to select the relevant language in the first step. In the next dialog with the basic settings (see Figure 3.21), the **Minimal Installation** option reduces the installation scope somewhat. After the installation, the complete basic system is available, but application programs like Thunderbird, LibreOffice, and an audio player are missing. These programs can be installed later if needed.

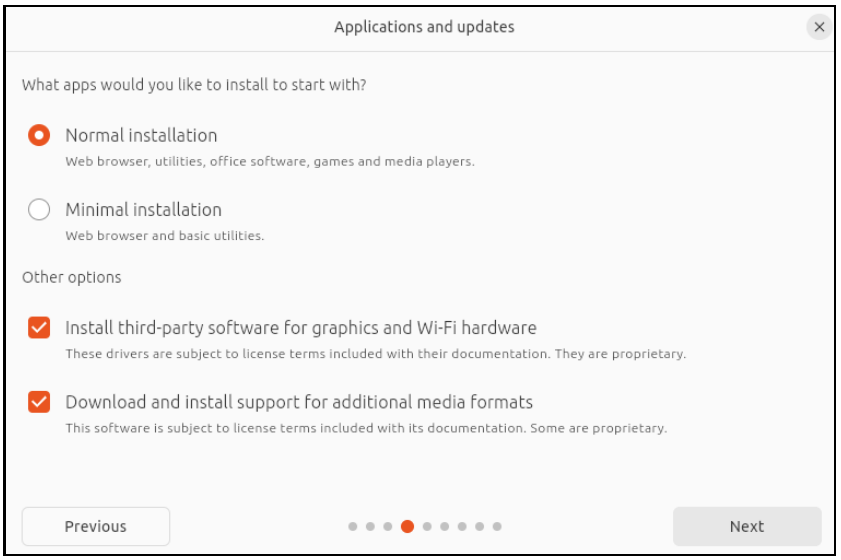


Figure 3.21 Basic Settings in Ubuntu Installer

The **Install Third-Party Software...** option controls whether various drivers that are not available as open-source software should be installed. Among other things, this affects the proprietary NVIDIA graphics drivers. If the installation program detects an NVIDIA GPU, the drivers are installed and are then already available at the first start.

Support for Additional Media Formats adds audio and video codecs to the installation. These codecs are necessary for the playback of video files to work.

Installation type

Depending on which other operating systems the installer detects on the SSD, the program then offers several installation variants to choose from (see Figure 3.22).

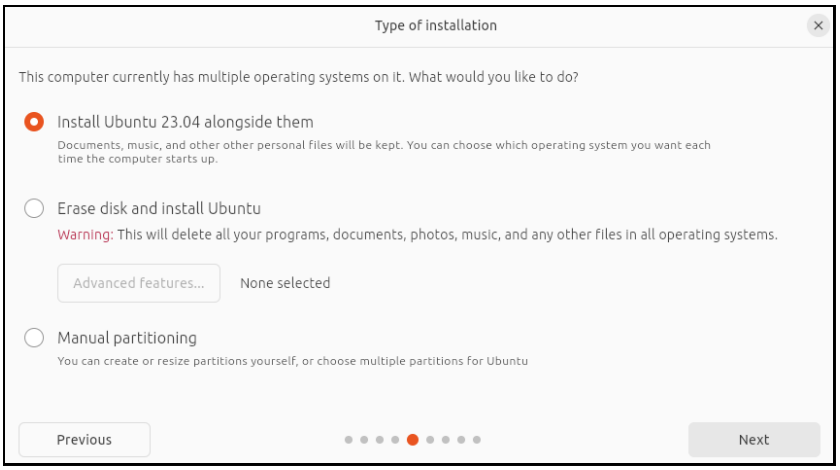


Figure 3.22 Setting Installation Type

The installer always shows only the options that are relevant for the computer at hand.

- **Install Ubuntu alongside Them**
  In this variant, the installer uses free areas on the SSD and installs Ubuntu there. After installation, you can select which operating system to boot from the EFI boot menu each time you reboot.
- **Erase Disk and Install Ubuntu**
  This will erase the entire hard drive or SSD and then repartition it.
- **Manual Partitioning**
  This option allows you to perform the partitioning yourself.

For most options (with the exception of **Manual Partitioning**), the installer creates a new system partition that fills the entire free space of the disk. You cannot influence the partitioning.

Manual partitioning

If you want to set the size of the partitions yourself, want your own `/home` partition, and so on, then you need to select **Manual Partitioning**.

To create a new partition, first select the **Free Space** entry and then click the plus button (**+**). In the dialog that appears, you must specify the type of partition, the size, and the file system (see Figure 3.23).

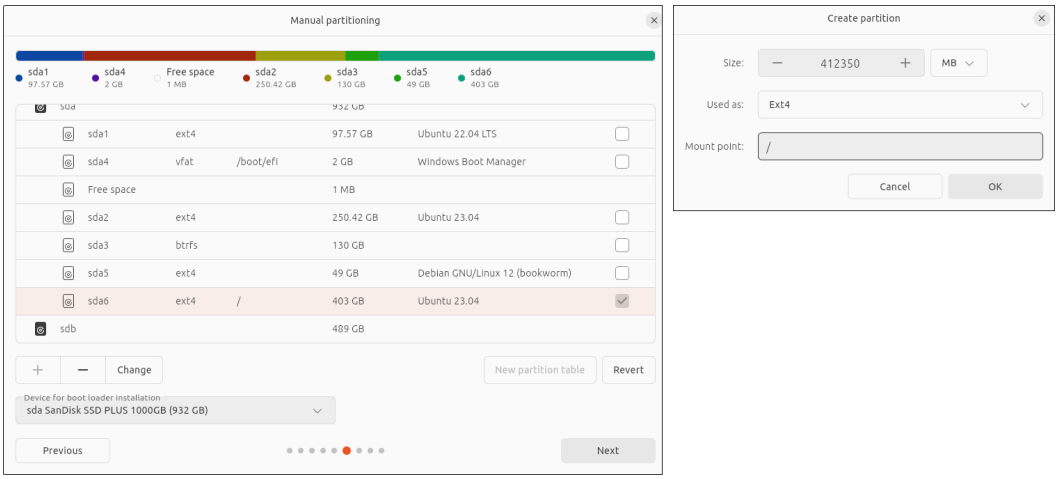


Figure 3.23 Manual Partitioning

If there is already a suitable partition on your hard disk where you want to install Ubuntu, select this partition, change the mount point (the mount directory), and, if necessary, activate the checkbox to reformat the partition.

You don't have to worry about the EFI system partition. As soon as you manually set up the first partition, the program automatically adds an ESP sized at 1 GiB.

After a final query, finish the partitioning by clicking the **Install Now** button. After that, you will not be able to stop the installation or change the partitioning.

Time zone

You can now make various remaining settings parallel to the installation. First, specify the time zone you're in. The installation program assumes that your computer's clock is set to the local time.

Configuring users

In the next step, specify the username and password for the first Ubuntu user. You can set up additional users later during operation if necessary. In contrast to other Linux installers, you do not need to specify a `root` password here: admin work can be done on Ubuntu by an ordinary user via `sudo`.

Desktop theme

In the last dialog, you can choose between a light and a dark appearance for the desktop. If necessary, you can revise this setting later during operation.

3.7.2 Getting Started

Performing updates

To bring all installed Ubuntu packages up to date, run the following command in a terminal window:

```
user$ sudo apt update
user$ sudo apt full-upgrade
```

ubuntu-restricted-extras

If you did not enable **Support for Additional Media Formats** during installation, then installing the `ubuntu-restricted-extras` package will make your Ubuntu system multimedia ready. Among other things, this installs codecs for all possible audio and video formats, including MP3:

```
user$ sudo apt install ubuntu-restricted-extras
```

Installing hardware drivers

Ubuntu provides proprietary hardware drivers for NVIDIA graphics cards as well as for some Wi-Fi adapters. To install them, open the **Applications & Update** program from the **Start** menu. The drivers suitable for your system are listed in the **Additional Drivers** dialog (shown in Figure 17.2 in Chapter 17). To activate, you must restart the computer.

Alternatively, the installation of such drivers can also be done in text mode, which is very convenient, especially when the graphics system doesn't work due to a lack of drivers. In this case, you first need to determine the list of available drivers using `ubuntu-drivers devices` and then install the recommended driver via `apt`:

```
root# ubuntu-drivers devices
vendor   : NVIDIA Corporation
model    : GP107GLM [Quadro P1000 Mobile]
...
driver   : nvidia-driver-470 - distro non-free
driver   : nvidia-driver-525 - distro non-free recommended
driver   : nvidia-driver-535 - distro non-free
driver   : xserver-xorg-video-nouveau - distro free builtin
root# apt install nvidia-driver-525
```

Kernel updates (LTS enablement stack)

In the course of the usual updates for Ubuntu LTS, you only get bug fixes for the kernel. However, the base version of the kernel remains unchanged by default throughout the lifetime of Ubuntu LTS. Things are a little different when you perform a fresh LTS installation: Canonical updates the installation media (Ubuntu 22.04.1, 22.04.2, etc.) about twice a year, sometimes

using more recent kernel versions. This increases the compatibility of Ubuntu LTS with new hardware.

But how do you get newer kernel versions on an old Ubuntu LTS installation? For this, you must explicitly install hardware enablement (HWE) packages. The following command applies to Ubuntu 22.04 LTS; for Ubuntu 24.04, you must adjust the version number accordingly:

```
root# sudo apt install --install-recommends linux-generic-hwe-22.04
```

Note, however, that this will put you into a rolling release mode for the kernel! When the next Ubuntu LTS update version is released (22.04.3, 22.04.4, etc.), you will again receive the then-current kernel as part of the usual updates!

PART II

Using Linux

Chapter 4
GNOME

On Windows or macOS, there is only *one* desktop environment, whose appearance and behavior only changes noticeably during version changes. On Linux, on the other hand, there are quite a few desktop systems to choose from:

- GNOME is the most popular desktop system. It's used by default in all enterprise distributions as well as many other distributions (e.g., Debian or Fedora). Some distributions use GNOME as a basis, but modify it to a greater or lesser extent. This is true for Ubuntu and Pop!_OS, for example. (The Cosmic Desktop of Pop!_OS currently consists of GNOME and a few extensions. In the future, however, Cosmic wants to break away from GNOME.)
- In addition, there are a number of distributions or distribution variants that are based on GNOME offshoots, such as Cinnamon or MATE. Even within the GNOME project, there is a variant called GNOME Classic Mode, which is visually and functionally oriented toward the ancient GNOME version 2.
- KDE is a desktop system for tech-savvy users. It offers significantly more configuration options than GNOME, which unfortunately comes with many confusing dialogs.
- Pantheon is the desktop system of the Elementary OS distribution. The goal of Pantheon developers is to recreate the simplicity and elegance of macOS on Linux.
- Xfce and LXDE are desktop systems specially optimized for computers with less powerful hardware. LXDE has recently been widely used on Raspberry Pis under the name PIXEL Desktop.

In this chapter, I will focus on GNOME, although I will also briefly discuss the Cinnamon and MATE variants. The next chapter is dedicated to KDE. Finally, some information about LXDE and its PIXEL Desktop variant can be found in Chapter 5.

Version numbers

GNOME used to have version numbers like 3.34, 3.36, or 3.38. Since 2021, a new numbering scheme has been in effect with integer version numbers that change every six months. This chapter describes GNOME version 44, released in spring 2023. Version 45 was released in the fall of 2023, version 46 is expected in the spring of 2024, and so on.

4.1 Personal Assessment

Which is the best desktop for Linux? The question does not arise with macOS or Windows as there is only *one* desktop predefined by the operating system. But with many Linux distributions, you have a choice among KDE, GNOME, Xfce, and others.

Personally, I work 90% of the time in GNOME. This desktop enjoys the widest distribution and the most development work goes into it. No Linux desktop is perfect, but GNOME is more stable than its alternatives. Basic functions such as launching programs, arranging windows, managing files, and changing settings are easier to use than in most other desktop systems.

Less is more

A central guiding principle in the further development of GNOME in recent years has been to focus on a few features and to make them as simple as possible to use. While other desktop environments give their users a plethora of options to choose from to control the look and function of the desktop, GNOME's motto is: Less is more!

The enterprise desktop

The development of GNOME is strongly promoted and shaped by Red Hat. Many design decisions reflect enterprise market requirements. Of course, this can be criticized. On the other hand, it's pleasing that some companies are still investing money in the further development of the Linux desktop. (Linux for the desktop is hard to make money from, but server applications are not.)

Usable only after comprehensive configuration

My preference for GNOME in no way means that I am completely satisfied with the program package. On the contrary: to me, it is incomprehensible why there is no minimize button in the window bar by default, why the dock is hidden by default even on large screens, why the dock is forcibly placed at the bottom of the screen, why there are no sensible ways to arrange multiple windows on the monitor, and so on. I have the feeling that GNOME is optimized exclusively for tiny notebook screens. However, my notebook is almost always connected to a large monitor, where GNOME completely fails in the default configuration. (Case in point: First you click the **Activities** button with the mouse in the upper-left-hand corner. Then the dock is displayed at the very bottom. To select an app, you then need to move the mouse pointer to the other end of the screen. This is absurd!)

Nevertheless, the reason I have become a GNOME supporter is its configurability through external tools and extensions. The *GNOME Tweak* program enables you to change some basic options hidden in the normal settings program. Also, by installing extensions, you can optimize GNOME in many ways specific to your preferences. Corresponding tips and instructions can be found in this chapter.

4.2 Getting Started

X or Wayland

Of course, GNOME runs in the graphics system! Currently, however, there are two implementation variants for the graphics system: the traditional X server (Xorg) and the newer Wayland system. I describe their differences in detail in Chapter 17.

GNOME is currently the only desktop system that is fully compatible with Wayland. With GNOME, most distributions therefore use Wayland by default, provided that suitable graphics drivers are available. The last major obstacle to the widespread use of Wayland is NVIDIA graphics cards, whose proprietary drivers are still not fully compatible with Wayland.

Ideally, it shouldn't matter which graphics system is active behind the scenes of GNOME. However, there are actually small differences, such as in the stepless scaling of the display on high-resolution monitors or in programs that want to read ("split") the screen content.

Many distributions therefore allow you to choose between the systems during login. In Fedora, the gear menu in the lower-right-hand corner of the login screen takes you to GNOME on Xorg (as shown in Figure 17.5 in Chapter 17).

Welcome wizard

Immediately after your first login to the GNOME desktop, a welcome wizard appears. There you can set the language as well as the keyboard layout, and you can connect GNOME to an online account, such as Google or NextCloud. However, you can do this step just as well later in the system settings.

4.2.1 Panel

The only permanently visible control element of the desktop is the panel, which is displayed immovably at the top of the screen (see Figure 4.1). It contains the **Activities** button, an icon for the currently active program, the time, and various status icons and menus on the right-hand side. The actual workspace is completely empty—if you disregard any open windows, that is.

Icons on the desktop

Displaying icons on the desktop is not intended. If you want that, you need to install an extension. On Ubuntu, this happens by default.

System menu

The icons on the far right of the panel take you to the system menu, where you can configure the network connections, adjust the volume, turn off the computer, log off, or switch to another user account.

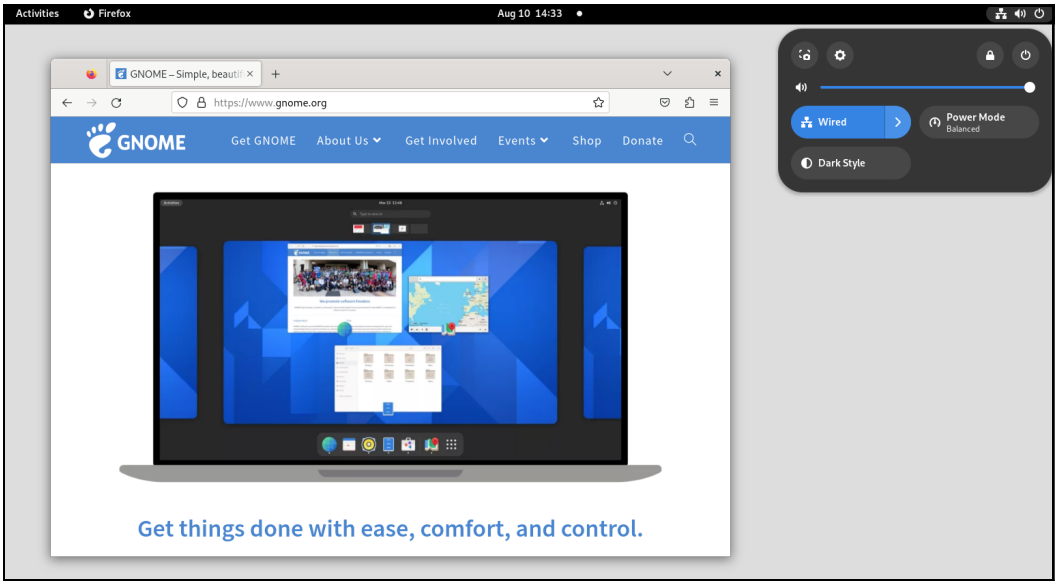


Figure 4.1 GNOME Desktop

Application menu

Some GNOME programs make their most important menu commands accessible via the so-called application menu. You can access this menu by clicking the name of the currently active program in the panel.

In the past, this menu has caused a lot of confusion. It was unclear which commands were hidden in the application menu and which were hidden in the actual menus of the program. The application menu may therefore disappear in future GNOME versions. Already now, many programs display only four entries: **New Window**, **Application Details** (which leads to the description of the program in the package management program), **Settings**, and **Exit**.

Panel icons

The icons displayed on the right of the panel indicating the network status, volume, and so on are predefined by GNOME. In addition, you can download and activate extensions from the *https://extensions.gnome.org* website (see Section 4.7). Some extensions integrate additional icons into the panel.

4.2.2 Activities

Clicking the dynamic workspace indicator (the button at the left side of the panel), moving the mouse cursor to the upper-left corner of the screen, or pressing the Windows key or Alt+F1 keys opens the *activities view* (see Figure 4.2). This view displays a horizontal overview of all workspaces at the top. In the main area of the screen, all windows of the active workspace are displayed in a kind of synopsis view. A dock appears at the bottom of the

screen with icons of frequently used programs and all running programs. In this view, you can change the active window and desktop, move windows to different desktops, reposition icons of frequently used programs in the dock, and so on.

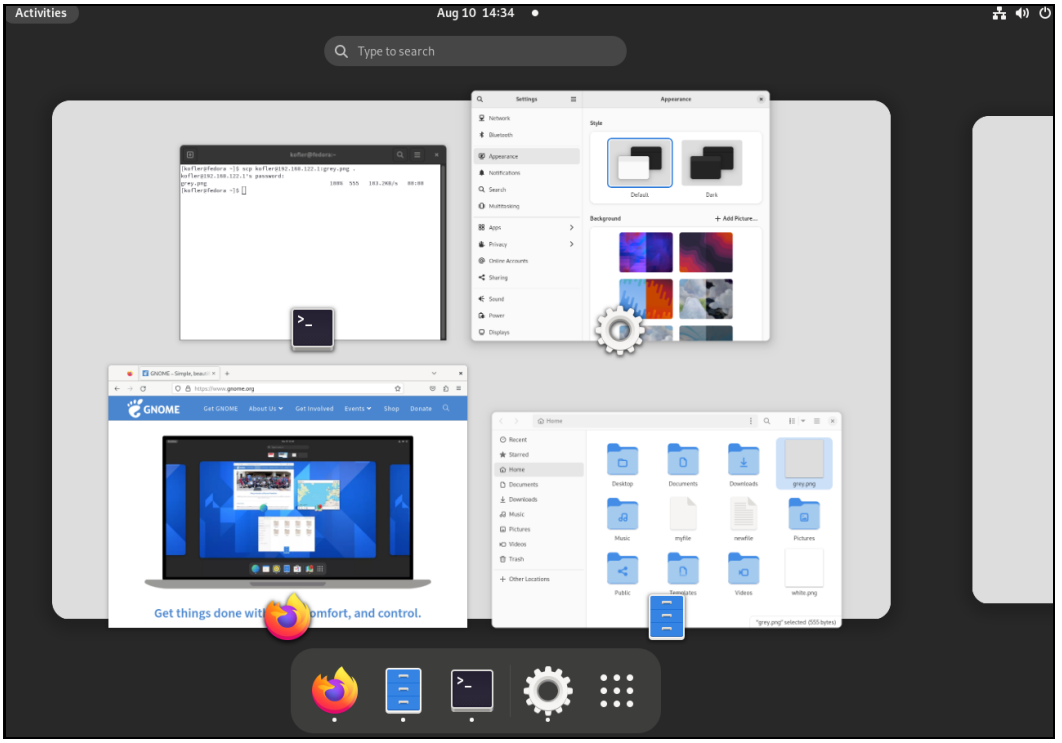


Figure 4.2 Activities View

Search function

A search field is active in the activities view. As soon as you enter a search term via the keyboard, GNOME replaces the synopsis view of all windows with the search results, taking into account programs, system settings modules, directories, contacts as well as the most recently used files. You can select the relevant object either using the mouse or the cursor keys.

The search function is an incredibly useful feature. For example, if you want to quickly open GIMP, just type [Windows] gi [Enter]. As soon as you get used to it and have memorized the first letters of the most important programs, starting the program this way will be extremely fast and efficient.

Note that [Windows] <name> [Enter] activates programs that are already running and does not start a new instance. This is usually expedient, but not always: for example, if you don't want to activate an already running terminal window but want to open a new terminal window, you need to press [Ctrl]+[Enter] or click the **Terminal** icon in the dock while pressing [Ctrl].

4.2.3 Dock (Dash)

In GNOME, there is no permanently visible taskbar or window bar. This role is played by the horizontal icon bar at the bottom of the activities view. The GNOME developers refer to the bar as *dash*, but I stick with the more common term *dock* in this book.

By default, the dock contains some programs that the GNOME programmers or the developers of your distribution assume you will need frequently. In addition, icons of all currently running programs are displayed in the dock if they are not included in it anyway. Currently running programs are highlighted.

To remove an icon from the first area of the dock, you want to run the **Remove from Favorites** command from the context menu. To add a program to the dock, you can drag and drop the program in question from the **Applications** view to the dock. Alternatively, if a program is already running, you can execute the **Add to Favorites** command from the context menu.

Left, Right, or Down? Visible All the Time or Only in the Activities View?

On Windows and macOS, it is common that the dock is always visible and can be placed on the left, right, or bottom edge of the screen. Unfortunately, the GNOME developers don't give you that much freedom. The dock is only visible in the activities view and only at the bottom of the screen. (Previous GNOME versions preferred the left margin, which seems more sensible to me—but none of the versions have provided the flexibility to configure where it appears.)

Fortunately, the rigid GNOME defaults can be circumvented by installing extensions (see Section 4.7). *Dash to Dock* is particularly popular in this regard. On Ubuntu and Manjaro Linux, this or a similar extension is installed by default.

4.2.4 Running Programs

Launching programs

To launch a program you don't know the name of, activate the activity view and click the **Show Applications** icon at the far right of the dock (or at the bottom in the case of a vertical dock). This way, or by pressing Windows twice, you reach an icon overview, which shows icons of all installed programs over several pages.

At first glance, the order of the programs in the program view seems as arbitrary as it is chaotic. As a matter of fact, it is the date and time of installation

that are taken into account here. For this reason, recently installed programs appear at the bottom of the last page.

Just like on a tablet or smartphone, you can arrange the icons yourself using the mouse or a trackpad and group them together. To move an icon from one page to the next, you must first drag it all the way to the left or right edge of the screen. If you make a little effort, the program view will be really clear at the end (see Figure 4.3).

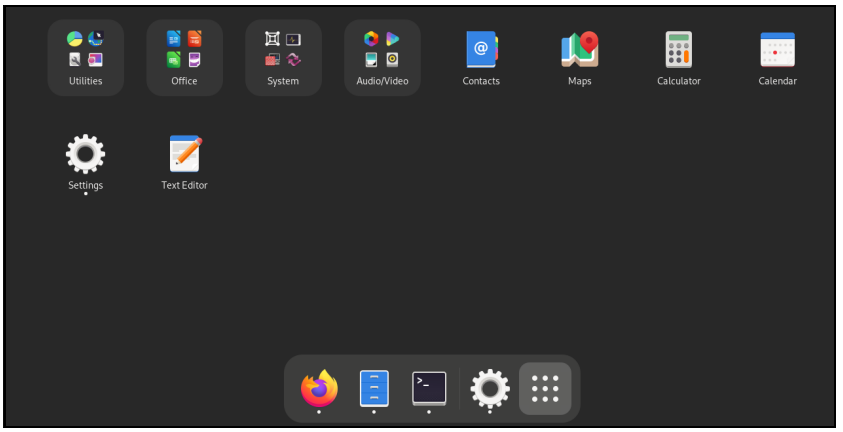


Figure 4.3 Program View after Extensive Cleanup

Alternatively, you can enter the name of the relevant program via your keyboard in the activities view. This is much faster and allows you to enter applications that haven't even been installed yet. GNOME will then display the **Software** program icon, which will help you install the respective application. This mechanism works only for desktop applications, not for other packages.

Windows

Due to an idiosyncratic design decision, GNOME lacks the **Minimize** and **Maximize** window buttons. To minimize a window, right-click the window bar and execute **Hide**; to maximize it, move it to the top of the screen or double-click the window bar. If you long for "normal" window buttons, it's best to perform the corresponding configuration by using the GNOME Tweak Tool. This program is described in Section 4.6.

As in Windows, you can place a window in the left or right half of the screen by moving it to the left or right edge of the window. If you then reduce or enlarge one of the two windows, the size of the other window will be adjusted accordingly.

Unfortunately, there are no comparable functions for quartering the window alignment (*quarter tiling*) or arranging it in other ways. Such functions would be useful for all users who work on large monitors. Windows 11

proves what can be possible in this respect. Fortunately, there are shell extensions for GNOME available to perform this task (Section 4.7).

Workspaces

Workspaces allow you to distribute the windows of running programs to several virtual desktops and to switch between them. This makes work easier and improves the overview when you open multiple windows at the same time. In the activities view, you can move windows to a second desktop. As soon as there are two active workspaces, GNOME provides a third, initially empty workspace. No matter how many workspaces you use, there is always one more.

For windows that are permanently needed, it is possible to mark them so that they are visible not only in one but in all workspaces. To do this, open the window menu using the right mouse button or Alt+Spacebar and activate the **Always in visible workspace** option.

To switch between the workspaces, you can use the Activity view or the keyboard shortcuts Ctrl+Alt+← or +→.

4.2.5 Special Features of Keyboard, Mouse, and Touchpad

Changing the active program

Using GNOME with the keyboard takes some getting used to. Alt+Tab switches not between windows as it does in Windows, but between programs. This concept is also common on macOS.

If a program consists of multiple windows or runs several instances at the same time (such as terminal windows), then you must select the desired window quite awkwardly using the cursor keys. There are two new keyboard shortcuts for this purpose: Alt+Esc switches between all windows, and Alt+^ switches between the windows of the currently active program. Thus, we now have *three* keyboard shortcuts to do what used to work beautifully with one shortcut.

Changing shortcuts

Fortunately, you can easily change all shortcuts in the system preferences in the **Keyboard** module if you don't agree with GNOME's default settings. The **Keyboard** module is the ultimate reference for all GNOME shortcuts; Table 4.1 summarizes only the very most important shortcuts.

Shortcut	Meaning
Windows	Switches between the standard view and the desktop overview (synopsis view). You can also get to this view by moving the mouse to the top-left-hand corner of the window. Then you can use the keyboard to enter search texts.

Table 4.1 Important GNOME Shortcuts

Shortcut	Meaning
Windows, Windows	Opens the program view.
Alt + F2	Starts the program whose name you specify.
Alt + Tab	Switches between programs (not windows!).
Alt + Esc	Switches between all windows (like Alt + Tab used to do).
Alt + ^	Switches between the windows within the currently active program.
Ctrl + Alt + Tab	Moves the input focus into the panel in the default view, allowing panel elements to be operated. In the desktop overview, Ctrl + Alt + Tab switches between different desktop elements: the panel, the dock (dash), the windows, the workspaces, and so on.
Ctrl + Alt + ← / →	Switches between the workspaces.
Shift + Ctrl + Alt + ← / →	Moves the current window to the next workspace.

Table 4.1 Important GNOME Shortcuts (Cont.)

As a desktop user, you work in graphics mode. But Linux also knows of so-called text consoles, between which you can switch by using special keyboard shortcuts (Ctrl + Alt + F1, + F2, etc.). However, this is only useful in exceptional cases—for instance, when the graphics system doesn't work properly after the notebook computer has been activated from sleep mode. Tips on using text consoles and a reference for the associated keyboard shortcuts follow in Chapter 6.

Using a mouse and touchpad

If you also press a button together with a mouse or touchpad click, then some practical additional functions are revealed:

- **Click+Ctrl in the dock**
  Opens another window of the already running program (useful for the terminal or the web browser, for example).
- **Click+Ctrl in the program view**
  Starts the program, but leaves the program view open.
- **Click+Windows in the window**
  Moves the entire window.
- **Mouse wheel+Windows**
  Switches between the workspaces.

Touchpad gestures

GNOME does not support as many touchpad gestures as macOS, but at least it's making progress in this regard as well:

- **Three fingers up**
  Activates first the activities view and then the programs view
- **Three fingers down**
  Returns to the normal desktop
- **Three fingers to the left/right**
  Switches to the next workspace

4.2.6 Using the Clipboard Efficiently

In almost all Linux programs with a graphical user interface, the same shortcuts apply when you use the clipboard as on Windows; thus, you can copy selected text using Ctrl+C or cut it using Ctrl+X, and then paste it using Ctrl+V. (In terminal windows, you must press Shift+Ctrl+C or Shift+Ctrl+V to copy and paste instead. *Caution:* Ctrl+C cancels the currently running command in the terminal.)

Selecting and pasting by mouse

Besides the traditional handling of the clipboard, in almost all Linux programs you can also copy text excerpts using the mouse and paste them at another place or in another program. To select text excerpts, simply move the mouse over the text while holding down the left mouse button. The text marked in this way is automatically copied to a buffer that acts like a second, implicit clipboard.

As soon as you press the middle mouse button or the mouse wheel, the text is inserted where the active input cursor is positioned. Selecting and copying is therefore done using the mouse only without the keyboard. Provided you have a mouse with a mouse wheel or a touchpad with three buttons, you will soon wonder why this doesn't work just as easily on Windows or macOS.

4.3 File Manager

The *Files* program is the GNOME file manager (see Figure 4.4). Not only does it grant access to files and directories, but it also allows you to access external media and network directories. This section describes the operation of the file manager but does not go into detail about the specifics of file management on Linux. In Chapter 9, you'll learn what links are, how hidden files are marked, how access rights work on Linux, and many other details.

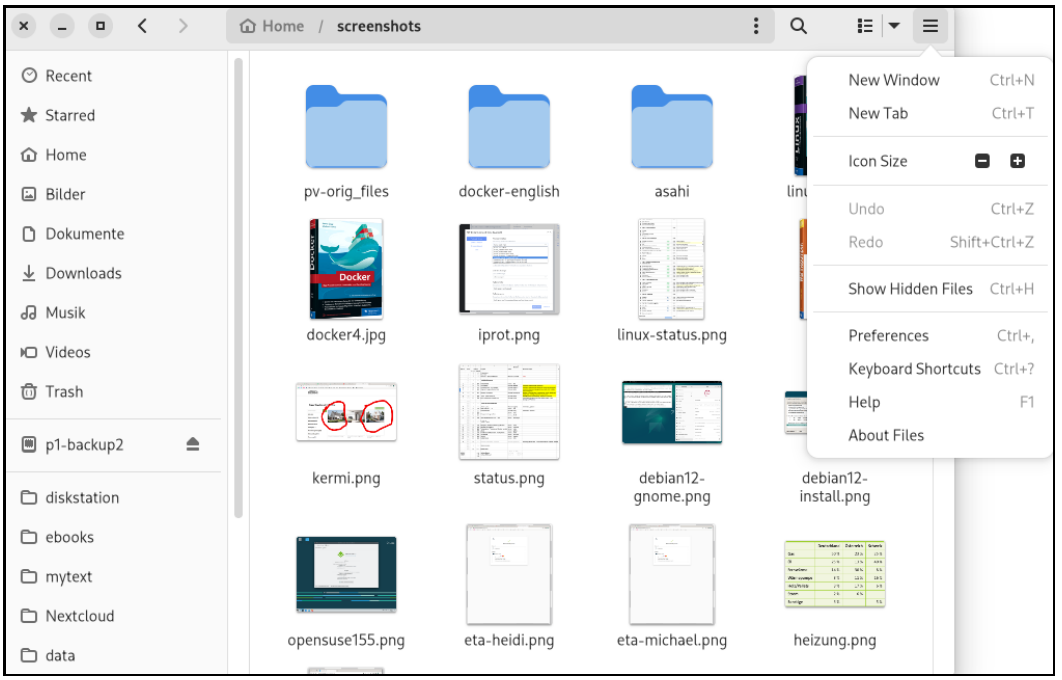


Figure 4.4 GNOME File Manager

Nautilus versus Files

In previous versions, the file manager was called *Nautilus*. Its new name is *Files*. This name change may be meant to be user-friendly, but since it isn't easy to form meaningful sentences with the new name ("Use Files to manage your files ..."), I will stick with the old name, Nautilus, or write *file manager* in this book. The name Nautilus has also been retained internally—for example, for the program file and for the package name.

4.3.1 Operation

Viewing files

The easiest way to start the file manager is to click its icon in the dock of the activities view. By default, the file manager displays the contents of the selected directory in the icon view. Each file is represented by an icon that, in the case of images and some other file types, provides a preview of the contents at the same time. By default, the preview function works only for local files up to 10 MiB. To make the preview work in network directories as well as for larger files, you need to change the corresponding options in the **Preview** dialog in the settings. You can access the settings dialog via the application menu in the panel.

Using Ctrl+1 and Ctrl+2 you can switch between the icon view and the details view. Within the icon view, you can use Ctrl++ and Ctrl+- to set the icon size.

The file manager saves the images in the `.cache/thumbnails` directory so that the preview does not have to be generated again and again. Many other GNOME programs also use this directory.

Displaying directories before files

By default, GNOME sorts all objects alphabetically, intermixing files and directories. If you prefer GNOME to show all directories first and then all files, enable the **Sort Folders before Files** option in the program settings.

> **Expandable Subdirectories**
>
> Nautilus can display the folders in expandable mode in the list view. This allows for easier navigation through the directory tree. This function can be activated in the **Display** dialog of the program settings.

Sidebar

The left edge of the window usually contains a sidebar that allows you to quickly switch to important directories. The upper area contains some predefined directories like `Images` or `Downloads`. This is followed by a section for external disks and network directories defined in `/etc/fstab`. In the lower area, you can drag and drop often needed directories and define bookmarks this way. Alternatively, Ctrl+D inserts the current directory as a bookmark in the sidebar. F9 switches the sidebar off or on again.

Changing directories

In the toolbar, the path to the currently active directory is represented by buttons. This allows you to quickly switch to parent directories.

Alternatively, the file manager displays the complete path at this point via Ctrl+L, which especially facilitates the quick entry of another directory.

Tabs

By pressing Ctrl+T, you can open a new dialog sheet. Dialog sheets are particularly useful when you want to copy or move files from one directory to another. During drag-and-drop operations, you can change the active dialog sheet.

Opening files

For most file types, double-clicking opens the file. The file manager will automatically launch a suitable program. If the file type is not known to the file manager, right-click the file and execute **Open with ...**. This will take you to a dialog that shows most of the programs installed on the computer for selection.

For some files, several programs are suitable for editing. For example, you can choose to open image files using an image viewer, GIMP, or Firefox. One of these programs is considered the default program and is started by double-clicking it. If you want to change the default program, right-click the

file, select **Open With ...**, select the program you want to use, and set the option **Always use for this file type**. After that, the setting will apply to all files with the same extension—for example, all `*.png` files.

Moving and copying files

Previously selected files can be copied via Ctrl+C or cut using Ctrl+X. You can then paste the respective files to the new location using Ctrl+V. Only now will the cut files be removed from their original location.

It's much easier to drag and drop files from one file manager window to another. In the process, the files are usually moved, not copied! An exception to this rule is drag-and-drop operations between different data storage media—for example, from a USB flash drive or from a network directory to the local file system. In such cases, a plus symbol is displayed in the mouse pointer.

If you want to copy a file instead of moving it, press the Ctrl key during the drag-and-drop operation. If you want to specify the move mode yourself, press the Alt key. After releasing the mouse, you will have the possibility to copy, move, or create a shortcut (link) to the file.

Finding files

The **Find** button allows you to enter a search term in the address field. The file manager then returns a list of all files that contain the search term in the file name. Following the search, you can limit the search results to a specific document type or directory.

Hidden files

On Linux, all files and directories whose names begin with a dot are considered hidden. This means that they are not normally displayed in the file manager or in file selection dialogs. Hidden files often contain configuration settings or other data that should not be changed directly. A direct editing of hidden files and directories is only useful in exceptional cases (e.g., if you want to back up your email directories in `.thunderbird`). To make such files and directories visible in the file manager, you need to run **Show Hidden Files** or press Ctrl+H (*h* for *hidden*).

Access rights

To ensure that not every user can read or change all files and directories, Linux stores the owner and access rights for every file and directory. The underlying concept is described in detail in Chapter 9, Section 9.7. To change the owner or access rights, right-click the file and run **Properties • Access Rights**.

Admin Access to the File System

By default, you can only modify your own files in Nautilus. However, to perform admin tasks, you can switch to a special root mode. To do this, press Ctrl+L and type `admin:` in the address bar. You will then be

prompted to enter your password. Provided your account has `sudo` privileges (see Chapter 10, Section 10.3), the file manager will switch to a root mode after successful authentication. This gives you unrestricted access to the entire file system. You should therefore work with appropriate care!

Renaming files

To rename multiple files, select the files and then run the **Rename** command from the context menu (see Figure 4.5). You can use the **Add** button to insert various name patterns—for example, to number multiple files.

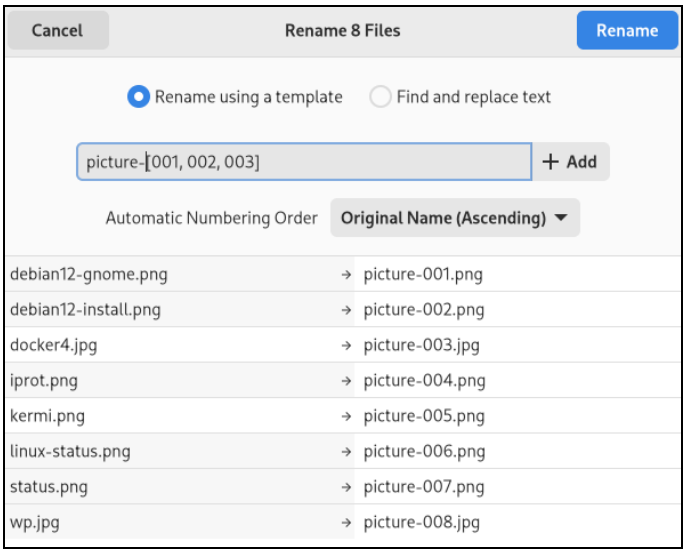


Figure 4.5 Renaming Multiple Files

Deleting files

You can press Del to delete previously selected files and directories. The affected files will initially end up in the trash (in the `.local/share/Trash` directory). You can see the contents of the trash by clicking the trash can icon in the sidebar of the file manager. Not until you empty the trash can will the files be permanently deleted.

Keyboard shortcuts

Almost all functions of the file manager can also be accessed via keyboard shortcuts. Fortunately, there is no need to memorize the abbreviations. Instead, you can run the **Keyboard Shortcuts** menu command: the file manager then displays a well-organized overview of all keyboard shortcuts.

4.3.2 Removable Media

When a USB flash drive or an external drive is plugged in, a new file manager window with the contents of the data media appears automatically. Remember that you must explicitly log off from external hard drives or

USB flash drives before disconnecting the cable from the computer! To do this, you need to click the **Eject** button in the file manager sidebar.

Formatting data media

To set up a file system on a USB flash drive or an external hard disk, you'll want to start the **Drives** program. This program also enables you to set up new partitions or to check the state of hard disks (see Figure 18.2 in Chapter 18).

4.3.3 Access to Network Directories

Windows shares

At the very bottom of the sidebar, the **Other Locations** item leads to a view that, after a few seconds, displays icons for removable media as well as for all devices detected on the network. The **Windows Network** icon is not useful here; double-clicking it shows active SMB1 network directories. However, this protocol is outdated, so the result is mostly **Folder Is Empty**.

You can establish a connection to a network device by double-clicking. If the network directory is password-protected, you must specify the login name and password. In the process, you will be allowed to store this data permanently in a GNOME password database. To avoid having to take the relatively laborious route to a network directory again and again, you can set up a bookmark using Ctrl+D.

Problems and their solutions

If the file manager can use a network directory without a password, it will automatically choose this variant. However, this approach is not always ideal: depending on how the Windows or Samba server is configured, the file manager will only show an empty directory afterward. There is now no way to log in by name via the user interface. To solve this problem, press Ctrl+L and insert your login name into the path. The correct syntax is `smb://loginname@servername/directoryname`.

If the file manager doesn't find any Windows servers, the most likely cause of the error is a too restrictive firewall between your computer and the Windows computer. On Fedora, RHEL, and SUSE, you must explicitly allow the Samba service by using `firewall-cmd` (see Chapter 29, Section 29.4). In many cases, only the name resolution does not work. To solve this problem, press Ctrl+L and type "smb://servername".

A number of other tips on using network directories on Linux follow in Chapter 27, Section 27.7. Note that access to Windows network directories works well in current GNOME versions but is abysmal in older versions.

Other releases and special views

Similar to the *smb://server/directory* notation, you can also establish connections to other server services; for example, `sftp://server` is particularly useful for accessing files on other Linux computers, provided an SSH server is active there. Table 4.2 summarizes the most important addresses and

protocols. In the table, you will also find three special addresses: `admin:`, `computer:`, and `trash:`.

Address	Result
`admin:`	Root access to file system (requires root login/ `sudo`)
`computer:`	List of all media
`afp://user@host/path`	Access to AFP server (Apple)
`dav://user@host/path`	Access to WebDAV server
`davs://user@host/path`	Access to WebDAV server (encrypted; i.e., HTTPS)
`ftp://hostname`	Access to FTP server
`network:`	Use as a general network browser
`sftp://hostname`	Access to SFTP server (SSH protocol)
`smb:`	Use as Windows network browser
`smb://hostname`	Access to Windows network directories
`trash:`	Trash can (deleted files)

Table 4.2 Special Addresses

Cloud storage from online services (such as Google) can also be displayed like directories in the sidebar. These services are configured in the **Online Accounts** module of the system settings. Note, however, that files stored in cloud directories cannot be edited directly by all programs. Double-clicking the file will then only result in an error message stating that the file could not be opened. To solve this problem, first copy the file to a local directory, then edit the file there.

GVFS Behind the scenes, the *GNOME Virtual File System* (GVFS) is responsible for accessing network directories. It includes external directories as subdirectories of `/run/user/<id>/gvfs` in the directory tree. The file selection dialogs of GNOME programs show external network directories in the sidebar (press F9 if necessary).

4.3.4 Sharing Network Directories

GNOME provides an amazing number of options to share your own data in the local network. Unfortunately, none of the variants works really well. If you read this section expecting that simply switching one or two buttons will lead to success, you will be disappointed. The fact that GNOME still fails

so miserably in this regard more than 20 years after the project's start is regrettable and a sign that Linux's desktop application is still in its infancy.

Sharing any directories via SMB

Both Windows and the Linux program Samba use the *Server Message Block* (SMB) protocol to share directories on the network. To share your own directory in the file manager on GNOME, you need to run **Sharing Options** from its context menu.

This menu command or the corresponding dialog sheet in the properties dialog is only available on Debian, SUSE, and Ubuntu, provided that the `nautilus-share` package with the plugin of the same name has been installed. You are out of luck with Fedora and RHEL; there, the `nautilus-share` package cannot be installed, presumably because it is incompatible with the SELinux configuration of these distributions.

If you set up the first of these shares on Debian or Ubuntu, the Samba program will first be installed after a confirmation prompt. On SUSE, you have to take care of this and a suitable firewall configuration yourself.

Shares without a password can be set up easily using `nautilus-share` (see Figure 4.6). If, on the other hand, you want to secure the directory with a password, you must then execute `smbpasswd -a your-login-name` in a terminal window and enter the relevant password there (but for security reasons, you must never use your regular login password!).

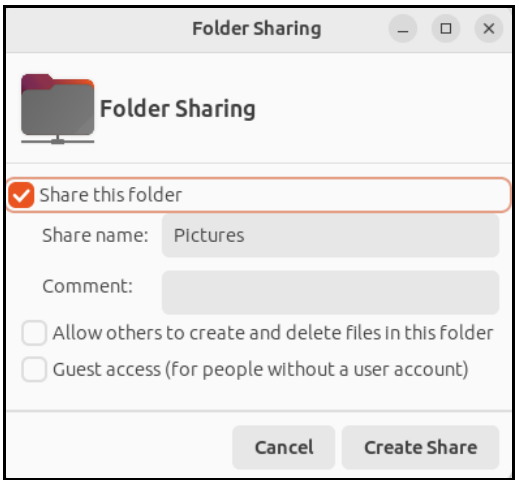


Figure 4.6 Network Sharing without Password on Ubuntu

See Chapter 27 for a lot of background information on manually sharing network directories. Since `nautilus-share` often creates more problems than it solves in practice, there is usually no way around the manual configuration of Samba anyway.

Shares in the system settings of GNOME

The **Sharing** module in the system settings provides an overview of various other sharing variants (see Figure 4.7). However, the number of options displayed there strongly depends on the additional software that is installed. The following list applies to Fedora 38:

- **File Sharing**
  File sharing via WebDAV; requires `gnome-user-share` package
- **Screen Sharing**
  Screen sharing via virtual network computing (VNC)
- **Media Sharing**
  Digital Living Network Alliance (DLNA) access to photos, videos, and audio files; `rygel` package
- **Remote Login**
  SSH login; `openssh-server` package

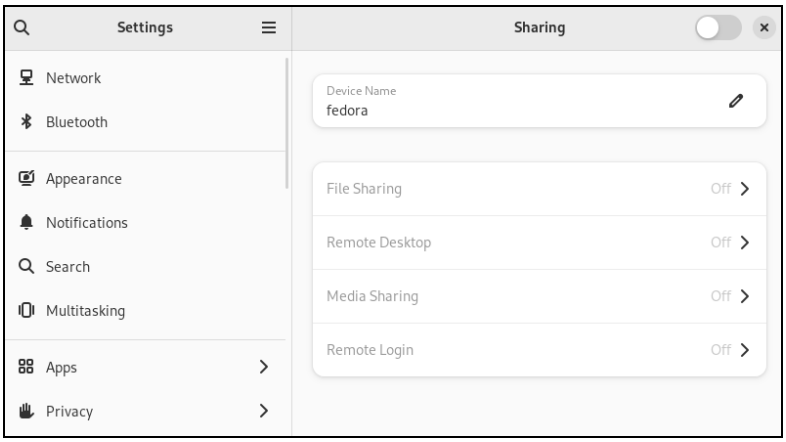


Figure 4.7 Sharing Variants in System Settings of GNOME

Some distributions require changes to the firewall configuration to use these sharing features. This concerns, for example, Fedora, Red Hat, and SUSE.

Sharing a public directory via WebDAV

Provided that `gnome-user-share` is installed, the **File Sharing** item opens a simple dialog in which you can share the `public` directory within your home directory using the WebDAV method on the local network—with or without a password. Unfortunately, this works extremely unreliably. I have tried this feature again and again over the past years—and it has always failed.

Sharing photos, videos, and audio files via DLNA

If the `rygel` media server is installed, you will find the **Media Sharing** entry in the **Sharing** module. This option enables you to share your `images`, `music`, and `videos` directories according to Digital Living Network Alliance (DLNA) guidelines with just a few clicks. Various manufacturers of multimedia

devices belong to the DLNA, which means that you can also display or play the shared files on such devices. In addition, some audio and video players are also able to discover the directories shared in this way. In a conventional file manager, however, the directories will not display.

Tips from the Field

To quickly copy individual files back and forth between my Linux computers, I usually use SSH. Each of my Linux machines runs an SSH server. Under this condition, you can easily type the address "sftp://name@host" in Nautilus using Ctrl+L. This will establish an SSH connection to the specified computer and display its home directory in Nautilus. For my purposes, this is often sufficient.

But this variant is probably too complicated for Linux beginners; moreover, it requires sharing the personal password to exchange data between different users.

An easy alternative can be to set up a directory on a NAS device for data exchange that can be read and written by everyone. Naturally, such a directory is unsuitable for security-relevant data. However, the process is well-suited for quickly sharing, say, your latest birthday photos without having to fiddle with a USB flash drive or taking an often-slow detour via the cloud.

4.3.5 Plugins

The file manager can be extended by plugins. However, many distributions do not have plugins installed by default. In your distribution's package management program, you need to search for "nautilus", install the packages you want, and then log back into GNOME. I present only a few selected packages here, where the package names apply to Debian and Ubuntu:

- `nautilus-image-converter`
  This plugin allows you to rotate or resize images via the context menu.
- `seahorse-nautilus`
  This plugin helps you encrypt the selected files via the context menu.
- `nautilus-dropbox`
  This Dropbox plugin helps you synchronize the `Dropbox` directory with your Dropbox account.

4.3.6 Additional Programs

Analyzing the hard disk usage

If you want to know in which of your directories the largest amounts of data are located, the *Analyze Disk Usage* program is a valuable aid (program

name `baobab`). The program shows in a descriptive graphic which directories and subdirectories contain what amounts of data (see Figure 4.8).

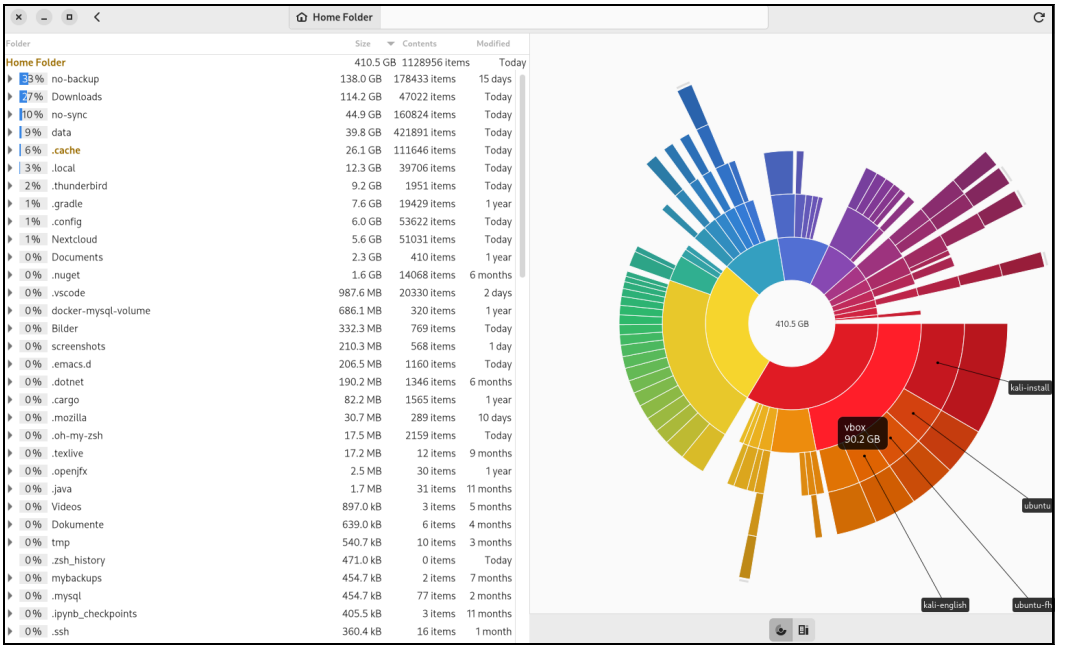


Figure 4.8 Displaying Space Requirements of Directories

ZIP files

You can unpack ZIP files and other archives in the file manager by double-clicking them. This creates new directories with the same names as the archive files.

Conversely, to create an archive, select the relevant files and then run the **Compress** command from the context menu. In the dialog that follows, you can then choose from several common archive formats (`*.zip`, `*.tar.xz`, etc.).

If you want to extract only single files from an archive or add more files to the archive, you can use the *Archive Manager* GNOME program for this purpose (program name `file-roller`). Although obsolete, this program can be easily installed in most distributions.

4.4 System Configuration

GNOME is not "only" a desktop environment. In fact, GNOME programs and especially the system settings (see Figure 4.9) help with many simple administration tasks, such as setting up a Wi-Fi connection or configuring

printers. This section introduces such preference modules and GNOME programs and cross-references chapters with further information and details.

Unfortunately, the modules seem to be ordered arbitrarily. So far, at least, I haven't been able to see any logic in the order of the modules, especially as the order keeps changing from version to version. You can use the **Find** button at the top left of the window to help locate a module!

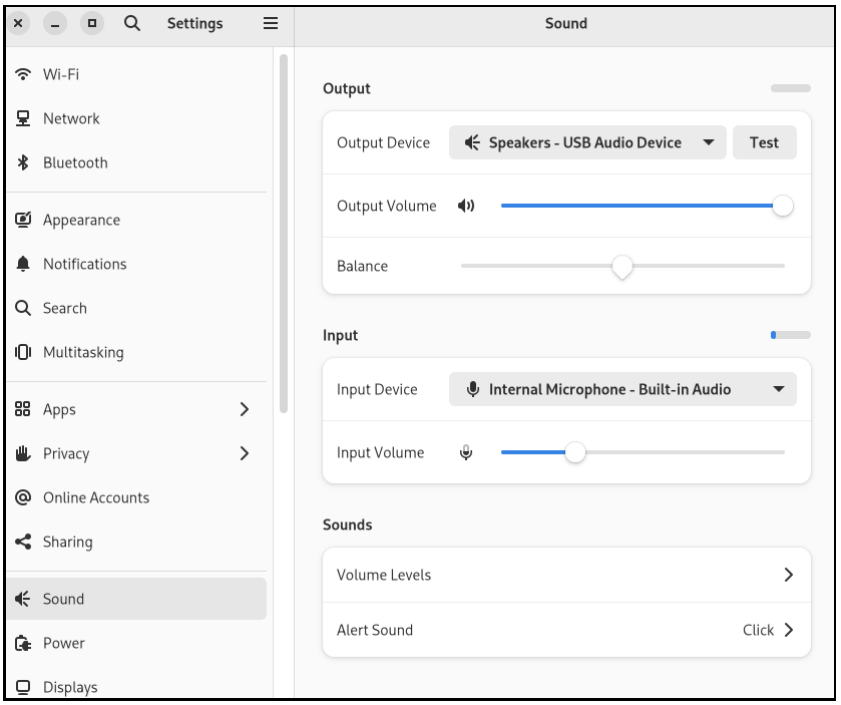


Figure 4.9 Overview of All System Settings Modules in GNOME

4.4.1 Mouse, Touchpad, and Keyboard

Keyboard

In the **Keyboard** module, you can set up multiple keyboard layouts. An icon then will automatically appear in the panel that allows you to change the currently active layout. In addition, the module allows you to list the numerous GNOME keyboard shortcuts and change them if necessary.

If you want to modify the functionality of the `CapsLock` key, swap `Alt` and `Ctrl`, or have other special requests, you need to run the `gnome-tweak-tool` program. (Some distributions require you to install the package of the same name first.) In its **Keyboard and Mouse** module, the **Additional Layout Options** button takes you to a dialog with countless control options (see Figure 4.10).

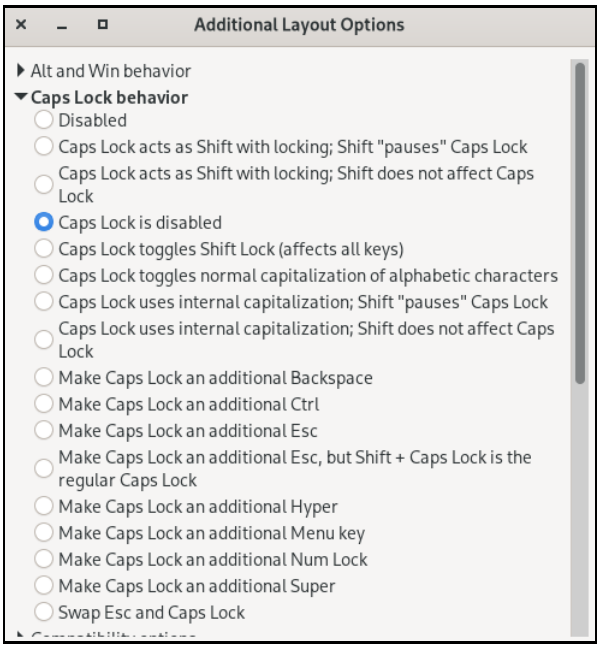


Figure 4.10 GNOME Tweak Tool to Modify Behavior of Special Keys

Note that the **Disabled** option displayed at the beginning of each group is misleading in this context. It does not mean that the key in question is disabled, but that there are no settings for the option. To actually disable the CapsLock key, you must select the **Caps Lock Is Disabled** option.

Mouse and touchpad

In the **Mouse and Touchpad** module of the GNOME preferences, you can swap the functions of the left and right mouse buttons, adjust the speed of the mouse or trackpad, and enable so-called natural scrolling. This enables you to move the window contents on the trackpad, rather than the scroll bar as you used to do.

As a rule, it's desirable to deactivate the touchpad of a notebook computer as soon as a mouse is available. Unfortunately, the GNOME settings do not provide a corresponding option for this, but you can achieve this goal by using the following command in the terminal:

```
user$ gsettings set org.gnome.desktop.peripherals.touchpad \
        send-events disabled-on-external-mouse
```

The command changes a hidden setting in the GNOME configuration database. You can restore the normal state, where the touchpad is always active, if necessary:

```
user$ gsettings set org.gnome.desktop.peripherals.touchpad \
        send-events enabled
```

4.4.2 Network Configuration

If your computer is connected to a router via an Ethernet cable, Linux will automatically establish the network connection itself. But the wireless network configuration is also very simple in most cases: You need to select the relevant wireless network in the system menu and enter the password once. Linux remembers the password and automatically establishes the network connection in the future as soon as your device is within the range of the Wi-Fi router. If you need to change a Wi-Fi password or have special (VPN) configuration requests, you will find corresponding setting options in the **Network** module of the system settings.

NetworkManager

Behind the scenes, the *NetworkManager* is responsible for the network configuration. This program is used by almost all Linux desktop distributions. I will introduce you to NetworkManager in more detail in Chapter 15. There you will also find comprehensive explanations of various technical terms and a detailed discussion of internal Linux network details.

4.4.3 Online Accounts

In the **Online Accounts** module, you can specify the login parameters of various online services, including Google, NextCloud, and Microsoft. These accounts can then be used in other applications, especially in the Contacts, Calendar, and Evolution GNOME programs.

Files in the cloud

If you save files in the cloud, Nautilus will show a new bookmark in the sidebar after configuring a corresponding online service. This bookmark leads to the cloud directory of the respective provider and makes it possible to copy files easily from or to the local file system.

Note, however, that integration into GNOME doesn't mean that an automatic synchronization with a specific directory takes place! Such functions are only provided by dedicated cloud clients. Suitable Linux programs are available for ownCloud, NextCloud, and Dropbox, among others.

4.4.4 Printers

Ideally, the printer configuration is done automatically: GNOME tries to detect USB, network, and Wi-Fi printers automatically and then performs the configuration on its own. A few seconds after the printer is connected or found on the network, the device is ready to print. It doesn't get any more convenient than this!

Unfortunately, this mechanism only works with relatively few printer models. As a rule, manual work is required. To do this, open the **Printers** module in the system settings. **Add Printer** then starts the search for printers. Your

printer may appear more than once in the list of found printers. In this case, there are different options (drivers) to address the printer. After selecting the model, the printer will be set up. GNOME may try to install a driver that exactly matches your printer model. Even if this fails, the printer can often be used with the help of a generic driver or a driver from another model. The **Printer Details** command leads to another dialog where you can select the desired driver from a long list (see Figure 4.11). For many laser printers, the generic drivers for PCL or PostScript models will work.

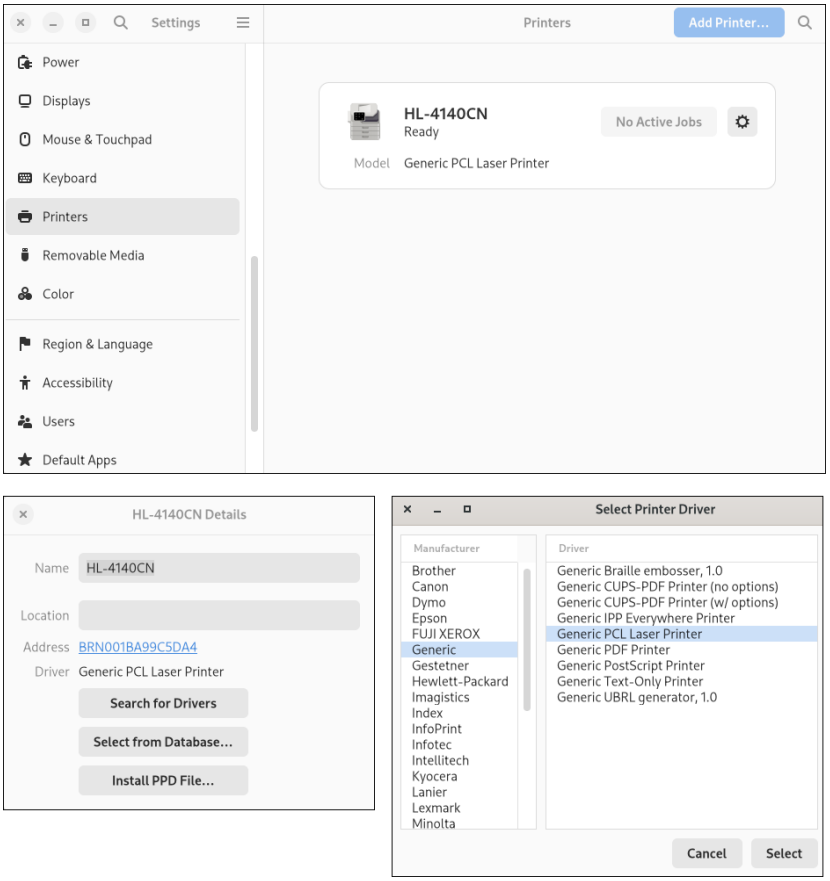


Figure 4.11 Printer Configuration on GNOME

system-config-printer

If this doesn't help you to configure your printer in the GNOME settings, you can alternatively try the old `system-config-printer` program, which can be installed optionally in some distributions. The program used to be very successful; however, it's no longer maintained and will therefore disappear in the long run.

CUPS

You can find a lot more details and internals about the Linux printing system, which is based on the Common Unix Printing System (CUPS) also used

on macOS, in Chapter 14, Section 14.11. There I will also demonstrate another configuration method via the CUPS web interface.

4.4.5 Monitor and Projector Configuration

In the **Displays** module or, depending on the GNOME version, **Display Devices**, you can change the desired screen resolution if you do not approve of the default resolution. If the screen then turns black, simply press Esc or wait for 15 seconds: the last valid configuration will then be restored. This is also the right place to set up a second display or a projector (see Figure 4.12).

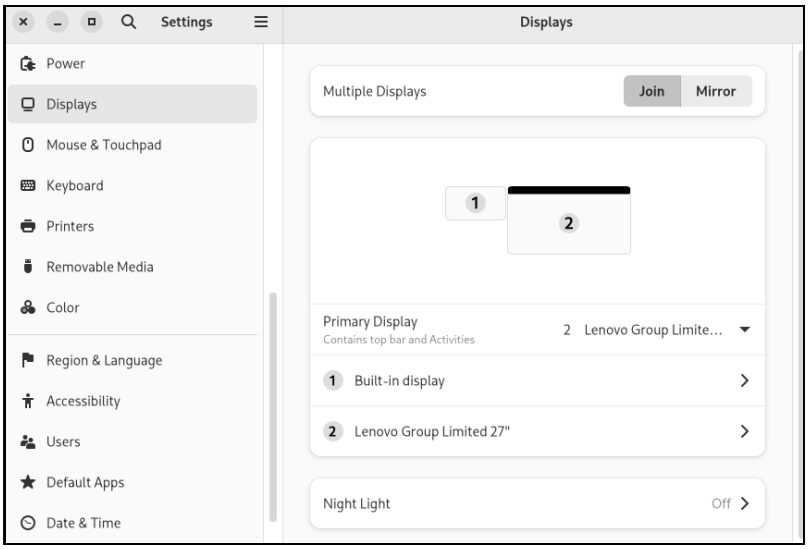


Figure 4.12 Configuring Computer with Two Displays

If you want to use multiple screens differently, you must mark one of them as the **Primary Display**. This is the screen where GNOME displays the panel and the dock. Unlike other desktop systems, GNOME is unfortunately not able to display the panel on all screens.

Contrary to outdated information on the internet, the GNOME settings can now also handle NVIDIA graphics cards (even when using the proprietary NVIDIA driver). However, the `nvidia-settings` program offers even more setting options and can also fulfill special requests.

Before you start `nvidia-settings`, you have to make sure that no monitor has been explicitly disabled in the GNOME settings. (The **Mirror Displays** setting is a good place to start.) Otherwise, it's possible that `nvidia-settings` won't want to recognize a monitor. Details on using the proprietary NVIDIA drivers follow in Chapter 17, Section 17.3.

GNOME saves the screen-specific settings in the `.config/monitors.xml` file and automatically activates them when you reconnect your computer to the same device. Chapter 17 contains a lot of background information on how the graphics system works.

Compatibility Issues

If you experiment with both graphics systems—that is, X and Wayland—then `.config/monitors.xml` may cause incompatibilities. If the changed graphics resolution suddenly doesn't work after switching between X and Wayland, you should simply delete the `monitors.xml` file and repeat the configuration.

4.4.6 High DPI Displays

Displays with particularly high resolutions (i.e., high DPI, 4K, 5K, or, in Apple jargon, Retina monitors) have proven to be a major problem for the Linux desktop in recent years. Support for such monitors on GNOME is good, though still not perfect.

Fractional scaling

Basically, you can switch between scaling levels—100%, 200%, and so on—in the settings. This works wonderfully, but it doesn't fit every monitor. A middle value between two of these settings would often be desirable. You'll obtain such a value as soon as you activate the **Fractional Scaling** option: this means that scaling factors 125%, 150%, and 175% are also available for selection. Unfortunately, the **Fractional Scaling** option is not always available, depending on whether your graphics system uses X or Wayland and how GNOME is preconfigured by your distribution. You have a good chance to have it work if you use Wayland on Ubuntu.

Behind the scenes, different mechanisms are used for fractional scaling:

- On X, `xrandr` creates a virtual display that has a higher resolution than the screen. The windows are output in this virtual display with double resolution. Finally, the image is scaled down to the actual screen resolution. This costs both computing power and image quality; nevertheless, the method works quite acceptably in real life.
- The procedure for Wayland-compatible programs is technologically better. In this case, each program takes care of the scaling of its windows itself. In the longer term, this technology will become established.

Whether your graphics system runs on Wayland or X, fractional scaling doesn't work equally well for every program; programs that use old libraries especially cause problems. Icons or menu texts, for example, are then displayed much too small in these programs.

4.4.7 Colors

The **Night Light** option in the screen settings causes the blue component of the monitor's light to be reduced after sunset. This is much easier on the eyes.

Color profiles

Graphic designers will give the night light mode and similar tools a wide berth. After all, you are concerned with displaying colors as realistically as possible on the monitor. You can define color profiles using the **Color** settings module.

4.4.8 User Administration

Account photo and password

In the **Users** module of the system settings, you can change your password and assign an image to your account by clicking your icon. You can either choose from among some standard images or use a photo from your `Images` directory.

Multiple users

The **Users** module also allows you to set up additional accounts—for example, for family members. To do this, you must first unlock the module by specifying your password or the root password (Debian, SUSE). Then you can add new users or change the password or rights of other users (see Figure 4.13).

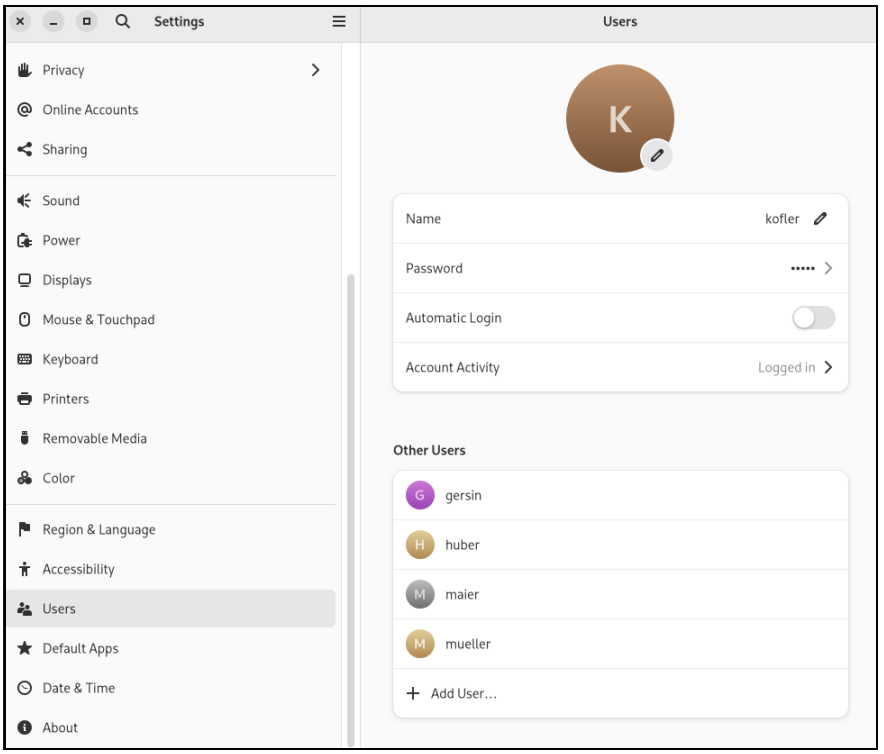


Figure 4.13 User Administration

System administrators are users who can gain admin privileges by specifying their own password using `sudo`. Details about this functionality are described in Chapter 10, Section 10.3. Only system admins are allowed to create or modify accounts of other users.

When setting up new users, GNOME offers the **Choose Password at Next Login** option. This means that the user initially has no password; he or she must set the password when logging in for the first time. This avoids the often cumbersome and insecure transfer of passwords.

Internal affairs

Comprehensive information about user and group management on Linux follows in Chapter 14, Section 14.5. The **User** module in GNOME is easy to use but only represents a fraction of the capabilities of Linux.

4.4.9 Software Installation and Updates

Once every few days, GNOME informs you that updates are available for the installed programs. Clicking the hint will launch the Software program and give you the opportunity to perform the updates immediately. Some distributions (such as Fedora) can also perform updates during a reboot if desired. Unlike Windows, however, a reboot is required much less often (especially after updating the kernel).

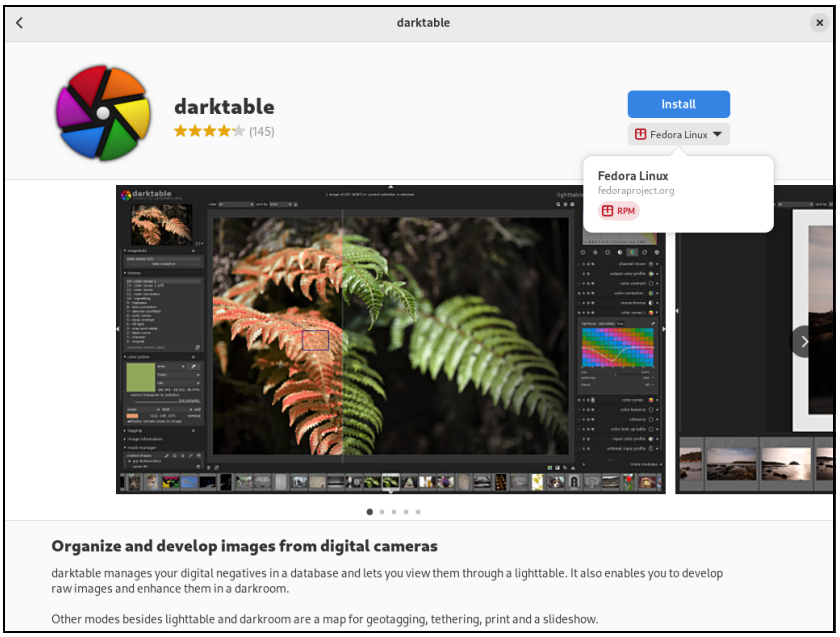


Figure 4.14 Installation of Additional Program

Software also allows you to search for and install other programs intended for your distribution (see Figure 4.14). Software is also eventually launched when you click a package appropriate for your distribution in a web browser. Instead of Software, several distributions use modified variants of this program (e.g., Ubuntu Software or Pop!_Shop), sometimes also proprietary developments.

Selecting the package source

Many programs are available multiple times in different package formats—on Fedora, for example, as an RPM package or Flatpak, or on Ubuntu as a DEB or Snap package.

In Chapter 16, Section 16.11, I will explain in detail why the traditional RPM and DEB package formats have recently been joined by completely new techniques for software management. You can set the required source in a menu in the top-right-hand corner of Software. My recommendation is to prefer the traditional RPM or DEB formats because their space requirements are smaller.

Software management by command

Advanced Linux users usually avoid using the Software program and install new software in a terminal window using the `apt` (Debian, Ubuntu), `dnf` (Fedora, RHEL), `pacman` (Arch Linux, Manjaro) or `zypper` (SUSE) commands. For a detailed presentation of these commands, see Chapter 16.

4.4.10 Remote Maintenance

When you need help, find the **Screen Sharing** button in the **Sharing** module of the settings. This button leads to a dialog where you can share your screen for remote control (see Figure 4.15). While remote maintenance was only compatible with the X graphics system in the past, the function can now also be used with Wayland.

The helper must then start a Virtual Network Computing (VNC) or Remote Desktop Protocol (RDP; commonly used in Windows) client. Common VNC clients on Linux include Vinagre and Remmina (both GNOME), `krdc` (KDE), TightVNC, and `vncviewer` from the `tigervnc-viewer` package.

Help Only within the Local Network

All VNC-based programs suffer from one major limitation: remote maintenance is only possible on the local network. Outside help usually fails because most computers are located in a private IP address space and are "invisible" to the outside world.

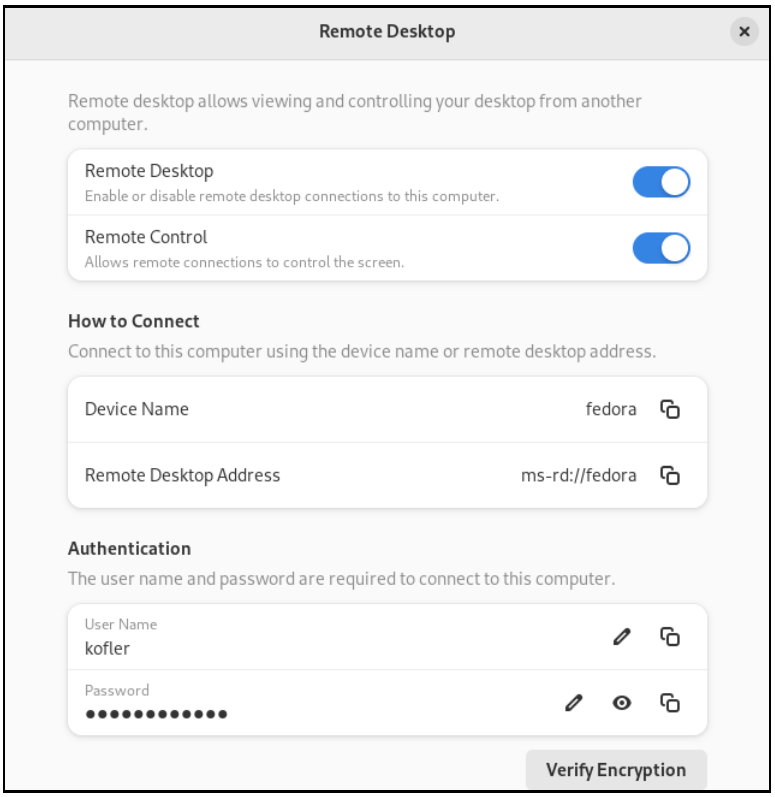


Figure 4.15 Setting Up Screen Sharing for Remote Maintenance

TeamViewer and AnyDesk

Attractive alternatives to screen sharing in GNOME include the commercial programs TeamViewer (*https://www.teamviewer.com/en-us/download/linux/*) and AnyDesk (*https://anydesk.com/en/downloads/linux*). They establish the connection in interaction with a server on the internet and can thus overcome the barriers of the local network. Private use is free of charge. Wayland is currently only supported experimentally by TeamViewer. With AnyDesk, the person seeking help must use X as the graphics system, while the helper must use Wayland.

4.5 Fonts

Installing fonts

Linux can basically handle all common font files—that is, TrueType, Type 1, and OpenType fonts. By default, quite a handsome selection of free fonts is already included. If you want to install more fonts, you have several options:

- You can search for font packages using your distribution's package management tools—but this isn't easy. While there are countless font packages available, many of them are only for specialized programs like LaTeX or for deprecated X applications.
- You can copy your own font files or fonts freely available on the internet to the `.fonts` directory. The fonts can then be used in all programs without further configuration. (You may have to log out and log in again beforehand.)
- By default, the `.fonts` directory does not exist. This can be quickly changed by opening a terminal window and running `mkdir .fonts`. In the GNOME file manager, the hidden directory initially remains invisible. Press Ctrl+H to show this type of directory.

Microsoft internet fonts

For a while, Microsoft used to offer various TrueType fonts for download (Andale Mono, Arial, Comic Sans, etc.). The fonts were supposed to enable all users to view web pages that use Microsoft fonts in optimal quality. The original download website no longer exists, but the fonts can now be downloaded from *http://corefonts.sourceforge.net*. The fonts may be used free of charge, but commercial distribution is prohibited. For this reason, the fonts are not included in commercial distributions.

Unfortunately, installing the fonts on Linux is tricky because the fonts are packaged in EXE files and may not be distributed in any other format. Installation instructions for distributions with RPM packages can be found at *http://corefonts.sourceforge.net*.

Depending on the distribution, there are scripts available to help you download and install the fonts:

- **Debian, Ubuntu**
  The `ttf-mscorefonts-installer` package contains the `update-ms-fonts` script. It installs the fonts into the `/usr/share/fonts/truetype/msttcorefonts` directory.
- **SUSE**
  The `fetchmsttfonts` package contains a script to download the fonts. You will find the font files in the `/usr/share/fonts/truetype` directory afterward.

Displaying and listing fonts

The *Fonts* program (program name `gnome-font-viewer`) displays all available fonts (see Figure 4.16). Clicking one of the icons shows various sample texts in this font. Alternatively, you can get a list of all fonts in a terminal window by using the `fc-list` command.

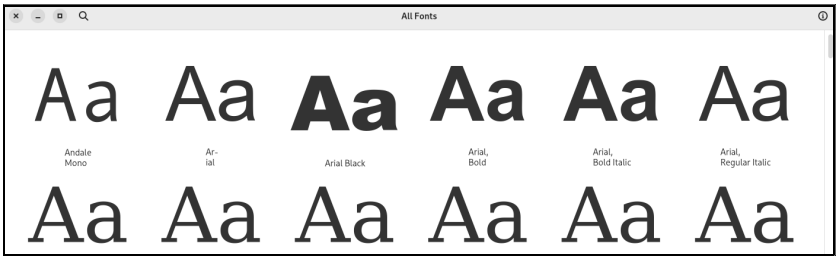


Figure 4.16 Overview of Installed Fonts

Character

The *Characters* program (internal name `gucharmap`) shows all available characters and allows you to copy special characters and emojis to the clipboard (see Figure 4.17).

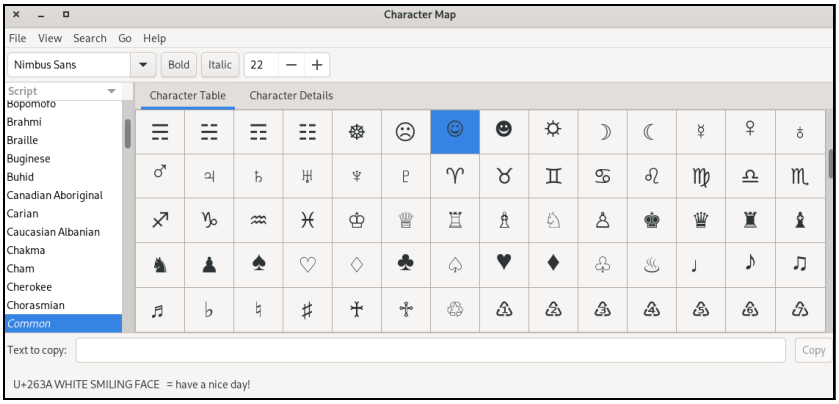


Figure 4.17 Display of Unicode Special Characters

Setting desktop fonts

You can use the GNOME Tweak Tool to set which font and size GNOME should use to display the desktop elements, menus, and window titles (see the following section).

4.6 GNOME Tweak Tool

The GNOME developers believe that it is not useful to give users too many options to change the look and feel of the desktop. Many Linux fans also associate Linux with the freedom to design the desktop according to their own ideas. The popular *GNOME Tweak Tool* (see Figure 4.18) helps with this. You can also start this program under the name *Optimization Tool*. If GNOME doesn't find the program, you must install it first. The package name is `gnome-tweak-tool`.

Among other things, the program allows you to set the following:

- Which buttons should be displayed in the window bar (if desired, also the buttons for minimizing and maximizing the window, which are missing by default)
- Whether the window buttons should be displayed on the right or left side of the title bar
- How GNOME should behave when the window bar is double-clicked
- Which functions special keys like `CapsLock` or the `Windows` key should have
- Which fonts should be used at what size on the desktop
- Which edge-smoothing methods should be used to display fonts (antialiasing, hinting)
- Whether the windows or the fonts should be scaled for high-resolution monitors (see Section 4.4)
- Whether GNOME should do without animations
- Whether modal dialogs may be moved freely on the screen (or whether they remain fixed to their base window, as is the default)
- Which themes and icons should be used to represent the desktop (see Section 4.8)
- Which programs should be executed automatically when GNOME is started
- How notebook computers should behave when the lid is closed
- Whether the file manager is allowed to display icons on the desktop
- Whether the date should be displayed together with the time

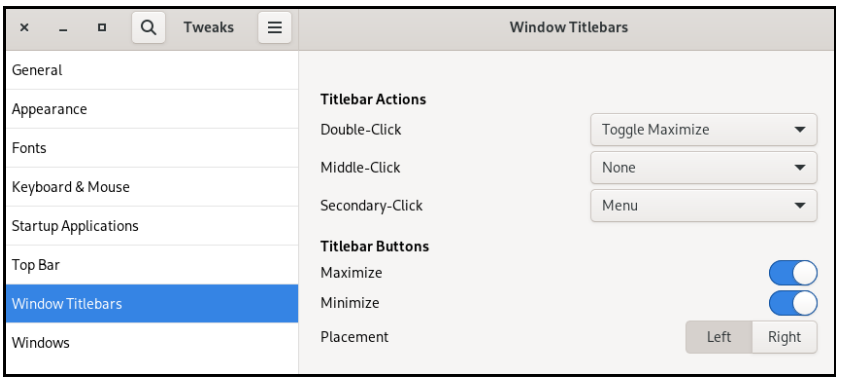


Figure 4.18 GNOME Tweak Tool Configuration Options

4.7 GNOME Shell Extensions

Beautiful design, but not perfect

When using GNOME, you get the impression that the development team had exactly *one* use case in mind: working on a notebook computer with a small monitor. Many design decisions are subordinated to the motto of saving space and using visual design elements sparingly. This has been quite successful: GNOME is easy to use and looks elegant.

I am less happy when using GNOME in the office on a 28-inch monitor. It isn't enough for me to be able to place the windows on the left or right of the screen when moving them: I'd like to quarter them or arrange them in an even finer grid by mouse or keystroke, as Windows 11 can do. And I would like to minimize windows I don't need at the moment via a button instead of having to switch to a context menu.

Extensions for the GNOME shell

When you read this, you may think, "These are strange wishes that Mr. Kofler has. I'm happy with GNOME as it is, but ..." And then comes exactly the one detail that bothers *you* about GNOME. But there's a way out of this dilemma: it's called *GNOME shell extension*.

The GNOME shell is the behind-the-scenes program responsible for managing the windows and displaying the panel and dock. This program makes extensive use of JavaScript. That's why it's possible to make comprehensive modifications to the desktop with just a few lines of JavaScript code. GNOME provides a special extension mechanism for this purpose.

You have three options to manage the shell extensions:

- Administration is possible using the relatively new *Extensions* program (program name `gnome-extensions-app`; see Figure 4.19).
- In the terminal, you can use the `gnome-extensions` command to list all installed extensions, reset or remove existing extensions, and so on:

```
user$ gnome-extensions list
appindicatorsupport@rgcjonas.gmail.com
arch-update@RaphaelRochet
awesome-tiles@velitasali.com
...
user$ gnome-extensions reset workspace-indicator@gnome-shell-
extensions...
```

- Alternatively, you can browse for extensions directly on the following website and activate them if needed (see Figure 4.20): *https://extensions.gnome.org*.

  I personally prefer this option, probably out of habit. However, controlling the shell extensions in the web browser requires that you enable an add-on when you first visit the page in the web browser and that you

also install the `gnome-browser-connector` package (formerly `chrome-gnome-shell`), if this is not already installed by default in your distribution.

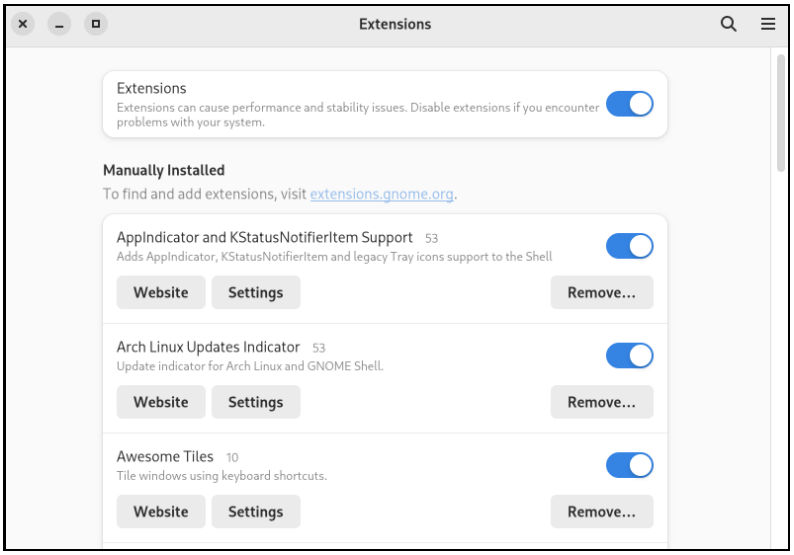


Figure 4.19 Administrating Shell Extensions Using gnome-extensions-app Program

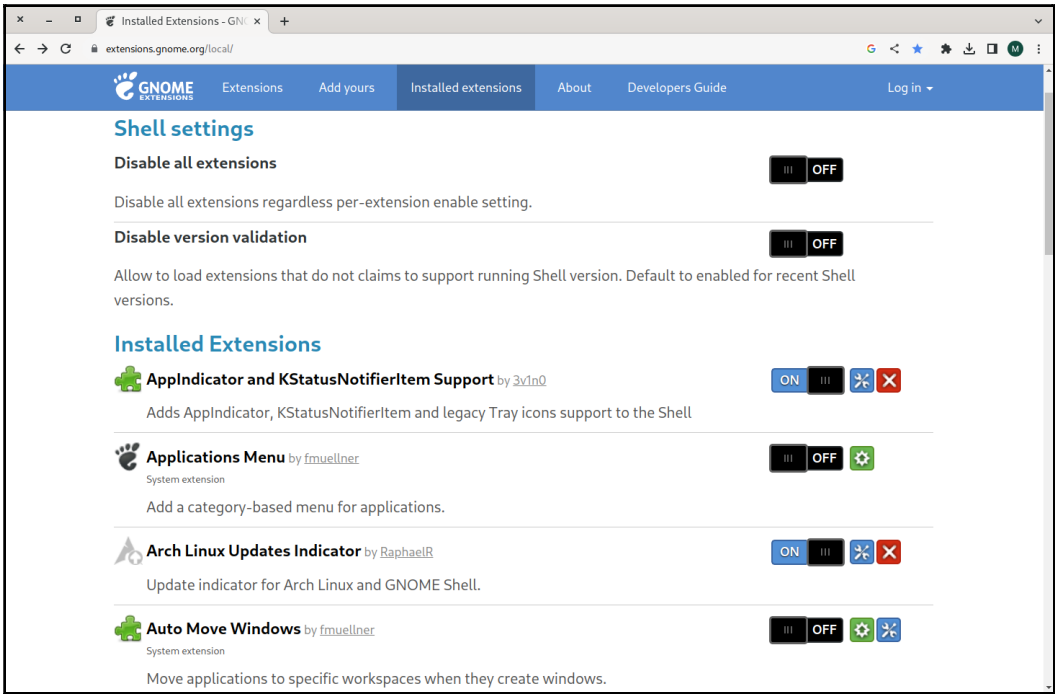


Figure 4.20 Searching, Activating, Configuring, and Updating GNOME Shell Extensions Directly in Web Browser

Installing shell extensions as packages

Some distributions provide important GNOME extensions as normal packages. You can use your distribution's package-management tools to search for and install them. For example, `dnf search shell-extension` provides an entire list of extensions with the official blessing of the Fedora developers. The `gnome-tweak` program again recommends (also on Fedora) to install the shell extensions as flatpaks.

I personally think that the administration of the extensions in the web browser is the simplest solution. Many extensions need to be updated after a GNOME update. In addition, many extensions are short-lived; every month, established extensions become obsolete while new ones are created.

Dream and Reality

Basically, shell extensions are a great mechanism to help GNOME become more flexible.

However, there is a risk of neglecting the actual work while spending hours on all of the fine-tuning. Besides, shell extensions inevitably cause trouble with every GNOME update. They will only work with the latest GNOME version when the developers update the extension as well. This can take weeks or months—unless further development has long stopped.

4.7.1 Useful Extensions

On the GNOME extension website, you can find real gems that make using GNOME much easier! I want to present some highlights here (in alphabetical order):

- **AppIndicator (KStatusNotifierItem)**
  This extension makes GNOME compatible with older programs that want to display a status icon in the panel.
- **Awesome Tiles**
  This extension makes it possible to place windows with custom shortcuts at predefined positions on the screen—for example, at the top left. I am a big fan of this extension because of its simplicity.
- **CPUFreq**
  This extension makes it possible to control the CPU speed and (de)activate the turbo boost mode. This mini program is especially useful for notebook computers.
- **Dash to Panel and Dash to Dock**
  Dash to Panel combines the panel and dock into one bar that can be placed on the left, right, top, or bottom of the screen as desired. In the

default configuration, the combined panel/dock looks similar to the taskbar in Windows. If you want, you can also leave the panel. In this case, the extension only implements a freely configurable dock.

Similar functions are offered by the *Dash to Dock* extension. In Ubuntu, a variant of this extension is active by default.

- **Desktop Icons NG (DING)**
  This extension allows you to display icons on the desktop. In previous GNOME versions, the file manager was responsible for this. However, this has no longer been the case for several years. This extension, which is active by default on Ubuntu, restores the lost function—ideal for all those who love an untidy, icon-covered desktop. :-)
- **GSConnect**
  This extension facilitates data exchange with Android smartphones.
- **gTile**
  This extension helps to place windows in a grid. This makes it easier to work on large screens.
- **MPRIS Indicator Button**
  This extension enables you to control many audio players via an icon in the panel. Unfortunately, this only works with restrictions for the Spotify program, because this audio player only incompletely supports the Media Player Remote Interfacing Specification.
- **system-monitor-next**
  This extension displays the CPU usage, memory usage, and other status information in the panel.
- **Tiling Assistant**
  This extension allows you to move windows into screen quarters, similar to Microsoft Windows. The move operations can also be assigned to keyboard shortcuts. It's possible that the extension will be delivered as standard in future Ubuntu versions.

4.7.2 Tips and Tricks

Installation path

All extensions are installed in the `.local/share/gnome-shell/extensions` directory. Note that too many extensions can affect the speed and stability of GNOME!

Synchronizing extensions

The more extensions you use, the more tedious it is to set them up again on a new machine or in a different distribution. You can save this work using the Extension Sync extension (see *https://www.omgubuntu.co.uk/sync-gnome-shell-extensions* and *https://extensions.gnome.org/extension/1486/extensions-sync*).

4.8 GNOME Shell Themes

GNOME shell themes

GNOME shell themes are files that modify the appearance of the desktop. Changeable elements include the color of the panel, menu and window borders, the design of buttons and other controls, and more. Many distributions make use of themes. On Ubuntu, Pop!_OS, Linux Mint, and others, GNOME doesn't look like it does on Fedora or Debian because of this. (The latter two distributions basically ship GNOME in the default configuration provided by the GNOME developers.)

Before you can use your own themes, you need to download the relevant theme from the internet and unpack it in a new subdirectory in `.themes`. If the `.themes` directory doesn't exist yet, you need to create it via `mkdir .themes` in a terminal window. The following site has established itself as a central collection point for countless themes: *https://www.gnome-look.org*.

Once the theme files are in a subdirectory of `.themes`, you can activate the theme in the GNOME Tweak Tool. To do this, open the **Appearance** dialog sheet and select the desired theme in the **Applications** item. (If the GNOME Tweak Tool is already running, you will need to restart the program so that it recognizes the theme.)

Icon themes

The icons used in the system settings and the dock can also be replaced with other images. At *https://www.gnome-look.org*, you can find icon sets for download. You must move the directory with the icons to the `.icons` folder after unpacking. When switching between the icon sets, the GNOME Tweak Tool helps again.

4.9 Internal Details of GNOME

dconf database

GNOME programs store their settings centrally in the `dconf` system. All `dconf` data is located in the binary database file, `.config/dconf/user`. GNOME programs access the `dconf` database directly via application programming interface (API) functions. If you want to read or change `dconf` settings from the outside, you should install the `dconf-editor` package and start the program of the same name (see Figure 4.21).

The `dconf` and `gsettings` commands allow you to change the `dconf` settings also in the terminal or by using a script. The following command causes Nautilus to use the list view by default, and not the icon view:

```
user$ gsettings set org.gnome.nautilus.preferences \
        default-folder-viewer 'list-view'
```

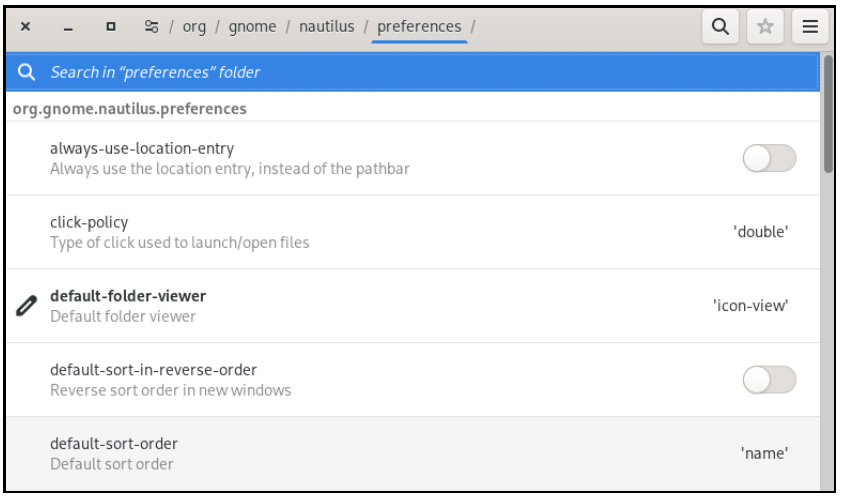


Figure 4.21 Changing Settings in dconf Database

Reset

If you've played around too much with `gsettings` or with `dconf-editor`, there is a pleasantly simple way to restore all settings back to their original state. To do this, open a terminal window and run the following command: `dconf reset -f /`.

Autostart

During GNOME startup, a lot of programs are started automatically. These are controlled by `*.desktop` files from the following Autostart directories:

`~/.config/autostart/*.desktop`	(personal Autostart programs)
`/usr/share/gnome/autostart/*.desktop`	(global Autostart programs for GNOME)
`/etc/xdg/autostart/*.desktop`	(global Autostart programs for all desktops; i.e., for GNOME and KDE)

There is no module in the system settings to control the automatic start of programs. In some distributions (such as Ubuntu), the *Startup Applications* program fills this gap (command name `gnome-session-properties`, `gnome-startup-applications` package). Alternatively, you can open the **Startup Applications** dialog sheet in the GNOME Tweak Tool.

The most tedious way is to set up the required `*.desktop` files yourself. The structure of such files is shown in the following example. This file is responsible for starting the Dropbox client:

```
[Desktop Entry]
Name=Dropbox
GenericName=File Synchronizer
```

```
Comment=Sync your files across computers and to the web
Exec=dropbox start -i
Terminal=false
Type=Application
Icon=dropbox
Categories=Network;FileTransfer;
StartupNotify=false
```

MIME

If Rhythmbox or Banshee appears automatically in Nautilus after double-clicking an MP3 file, then GNOME's MIME settings are responsible for this. MIME stands for *Multipurpose Internet Mail Extensions* and is a kind of database that creates a mapping between file types and programs.

The easiest way to make changes to the MIME configuration is to do it directly in the file manager: you click on the file, select **Properties • Open with** from the context menu, and choose the relevant program. The setting will apply to all files with the same extension in the future.

Individual changes to the MIME configuration are saved here:

```
~/.local/share/mime/*
~/.local/share/applications/mimeapps.list
```

For more information about the MIME database on GNOME, see *https://standards.freedesktop.org/shared-mime-info-spec/shared-mime-info-spec-latest.html*.

4.9.1 XDG Directories and Scripts

Several years ago, a set of common standards was defined as part of the Portland Project. They help to properly integrate programs into the desktop independently of GNOME or KDE. Later, the X Desktop Group (XDG) continued this work; today it's known instead as the *freedesktop.org* project.

Default directories

At first login, the `Images`, `Documents`, `Downloads`, `Music`, `Public`, `Videos`, and `Templates` subdirectories are created in the home directory. If a language other than English is set, these directories are given different names. Behind the scenes, the `xdg-user-dirs` package is responsible for the directories.

The configuration is carried out via the `.config/user-dirs.dirs` file. This file ensures that XDG-compatible programs find the directories regardless of the language that has been set. In GNOME, the directories are even renamed accordingly after a query if the language has been changed (`xdg-user-dirs-gtk` package).

If you don't want the default directories, you can delete them and create the following new file:

```
# ~/.config/user-dirs.conf
enabled=False
```

You can make this setting for the entire system in `/etc/xdg/user-dirs.conf`.

Configuration directories

Many, but unfortunately not all, GNOME and KDE programs use specially designated directories to save configuration settings and internal data. The directory names start with a dot and are thus considered "hidden". These directories therefore are not displayed in the file manager by default:

- The `.cache` directory is intended for storing temporary files that can be regenerated when needed, such as reduced images (thumbnails), search indexes, and so on. Caching is used to speed up frequently occurring workflows.
- The `.config` directory is intended for storing program settings, with each program using its own subdirectory.
- User data is stored in the `.local` directory. Usually each program creates the `share/programname` subdirectory for this purpose.

xdg scripts

The `xdg-utils` package provides the following scripts. A more detailed description can be found in the `man` pages of the respective commands:

- `xdg-desktop-menu` adds a new entry to the desktop menu.
- `xdg-desktop-icon` installs a new icon on the desktop.
- `xdg-icon-resource` installs icon resources.
- `xdg-mime` queries the MIME database or sets up a MIME data type.
- `xdg-open` opens a document in the user's default program. The command is extremely useful for opening a document, an image, or a video from the terminal—for example, `xdg-open picture.jpg`. In some distributions, the `open` alias exists for `xdg-open`, which saves some typing effort.
- `xdg-email` sends an email in the user's default email program.
- `xdg-screensaver` controls the screensaver.

4.10 GNOME Classic

Many Linux users have found it difficult to switch from GNOME 2 to the current versions. Especially for this user group, the GNOME developers have created GNOME Classic. This is a predefined collection of GNOME extensions that make today's GNOME look similar to GNOME 2. There is a

traditional start menu, icons on the desktop, and a taskbar at the bottom of the screen (see Figure 4.22).

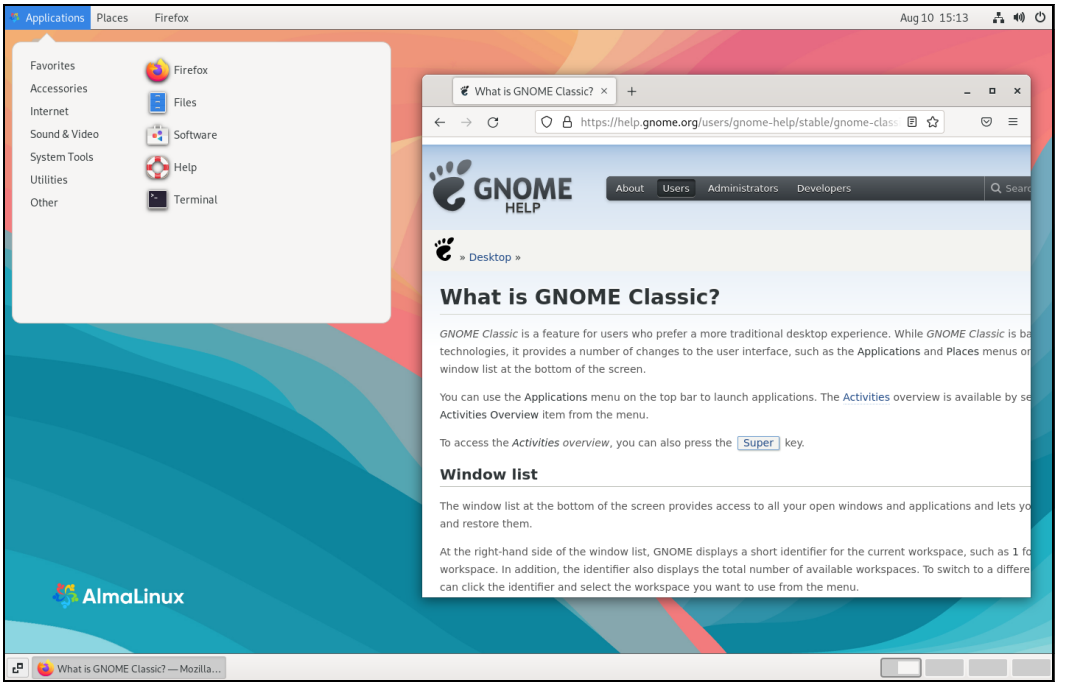


Figure 4.22 AlmaLinux in Optional GNOME Classic with Traditional Start Menu

In RHEL, the ordinary GNOME shell is normally used, but the packages required for the classic mode are still installed. When logging in, enterprise customers have a choice between the modern GNOME shell or the traditional GNOME appearance.

For other distributions, the classic mode extension packages must be installed separately. The package name is often `gnome-classic-session`. The next time you log in, you can then choose between **GNOME** and **GNOME Classic**.

Classic mode doesn't offer true compatibility with GNOME 2. In particular, the applets familiar from GNOME 2 are not available. There is also little room for maneuvering in the configuration of the two panels. Personally, I would have liked to have seen configurable quick-launch icons in the panel, for example. However, these can be implemented using the Frippery Panel Favorites GNOME extension if required.

Chapter 5
KDE

After the detailed introduction to GNOME in the previous chapter, I will introduce you to the most important alternative here: KDE. KDE basically performs the same tasks as GNOME, but it looks different and uses completely different internal libraries and protocols. While the objective of GNOME is to *keep it simple*, KDE offers its users maximum configurability and therefore tolerates confusing dialogs and nested menus. In this respect, KDE is aimed at technically oriented Linux users who want to design their working environment unrestricted by Gnome guidelines.

The prejudice that KDE is more difficult to use than GNOME, on the other hand, no longer applies. Nobody forces you to use the various special features of KDE. If you simply leave the basic settings, you will get a visually and functionally appealing desktop whose operation is not fundamentally different from what you are used to from macOS, Windows, or GNOME.

The focus of this chapter is on the basic features of KDE. Note, however, that the layout of the login screen, the start menu items, the visual design of the desktop, and the selection of included programs and configuration utilities may vary a bit depending on the distribution.

Free Choice of Apps

Each distribution gives you a mix of applications optimized for the desktop system you choose. These include a file manager, a terminal program, audio and video players, and so on.

This predetermined selection does not restrict you in any way! You can install and use the more powerful KDE counterpart Okular on GNOME instead of the Evince PDF viewer, for example. And in KDE, you can install the much simpler Shotwell counterpart to the digiKam photo management program, which has been optimized with countless functions for professionals.

5.1 Basic Principles

KDE originally stood for *Kool Desktop Environment*; later it became *K Desktop Environment*. KDE is based on *Qt*, an open-source library originally developed by Troll Tech. The strengths of Qt lie in its support for numerous

Qt library

platforms. Programs developed with Qt can therefore also be run on Windows and macOS. Comprehensive information about KDE and Qt can be found on the following websites:

- *https://kde.org*
- *https://www.qt.io*

5.1.1 Terminology

When I write *KDE* in this book, it's shorthand. The *KDE community* is a group of developers who develop various KDE software products—in their entirety, the *KDE Software Compilation*. This consists, among other things, of the various *KDE applications*, the *KDE Frameworks* (the libraries), and *Plasma* (the KDE desktop).

Of course, each of these components has its own version number. In this respect, it's impossible to talk about *the* KDE version number. When I wrote this chapter, the following versions were the latest:

- KDE applications: 22.12
- KDE Frameworks: 5.107
- KDE Plasma: 5.27

Currently, the KDE project is working on a migration to version 6 of the Qt library. As of February 2024, KDE Plasma and KDE Frameworks have made the leap to version 6.

5.1.2 Distributions

Most major distributions support KDE, either as a derivative or spin (Kubuntu, Fedora KDE Spin), or as an option during installation (Arch Linux, Debian, openSUSE). I used Fedora, KDE Neon (discussed next) and openSUSE for this chapter.

KDE Neon

One appealing distribution for KDE fans is *KDE Neon*. The stable foundation of this distribution is Ubuntu LTS. The KDE packages, on the other hand, are always up to date. Regarding KDE, Neon follows the rolling release approach: a few days after a new KDE version is ready, the corresponding packages are already available as updates for Neon. So if you choose Neon, you will always be up to date with regard to KDE.

Within the Neon family, there are again four variants:

- I recommend the *User Edition*, where you always get the latest KDE versions as soon as they are officially released by the developers.

- With the *Testing Edition* and the *Unstable Edition*, you always get daily updated packages that are still under development. Errors are to be expected to a greater extent.
- The *Developer Edition Stable* corresponds to the *Unstable Edition*, with various additional developer packages installed. As the name suggests, this variant is aimed at programmers who want to contribute to the KDE project.

5.2 Operation

KDE and the mouse

Perhaps the most obvious difference between KDE and other user interfaces is the use of the mouse: in KDE, instead of a double-click, a single mouse click is sufficient to open files, start modules or perform comparable operations. This takes some getting used to at first, but it allows for efficient and comfortable work.

If you don't want to change, you can of course set up KDE to be double-click compliant. To do this, start the **System Settings** in the KDE menu, go to the **Workspace Behavior** module, and activate the **Click Files or Folders = Selects Them** option. Fedora's KDE spin is configured this way by default, and KDE Plasma version 6 and later will also have this setting by default.

> **Applying Settings**
>
> A major difference between KDE and GNOME is that you have to explicitly confirm each change in a configuration dialog by applying it in order for it to take effect.

Structure of the desktop

Figure 5.1 shows the KDE desktop. By default, it's composed of a panel at the bottom of the screen and the actual workspace. The panel contains the KDE menu, optionally a few icons for the quick start of programs, a taskbar with icons of all open windows, and various utilities.

Plasmoids

The actual workspace (desktop) is usually empty at first. You can run widgets directly in the desktop or in the panel; these widgets are called *plasmoids* in the KDE terminology. The **Add Widgets** command from the context menu enables you to add plasmoids to the desktop (see Figure 5.2). In this context, you have a rich choice of clocks, status indicators (network usage, storage space, etc.), helper functions, and so on.

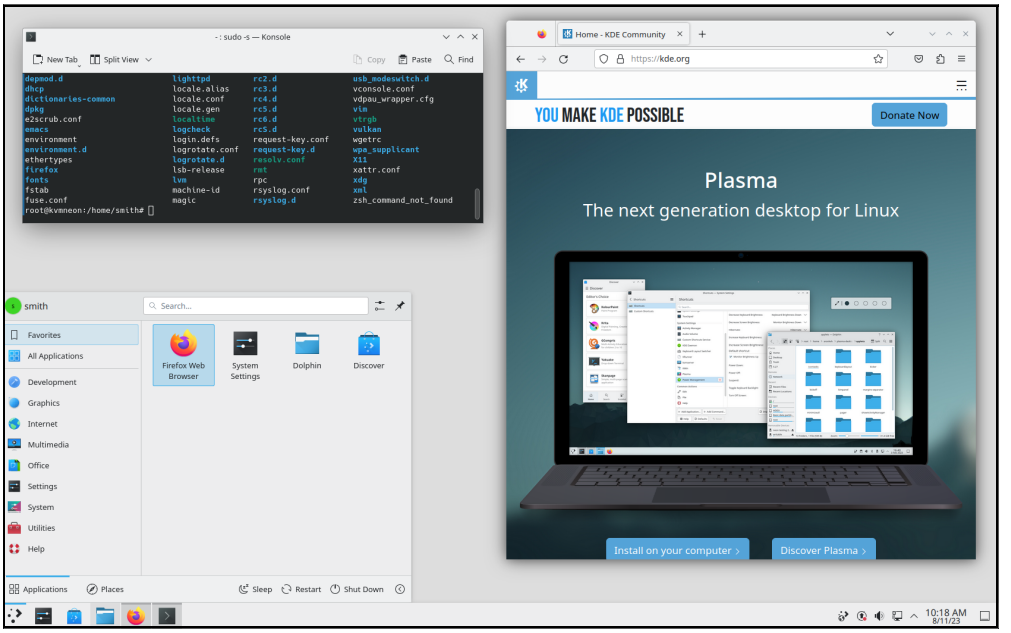


Figure 5.1 KDE Desktop

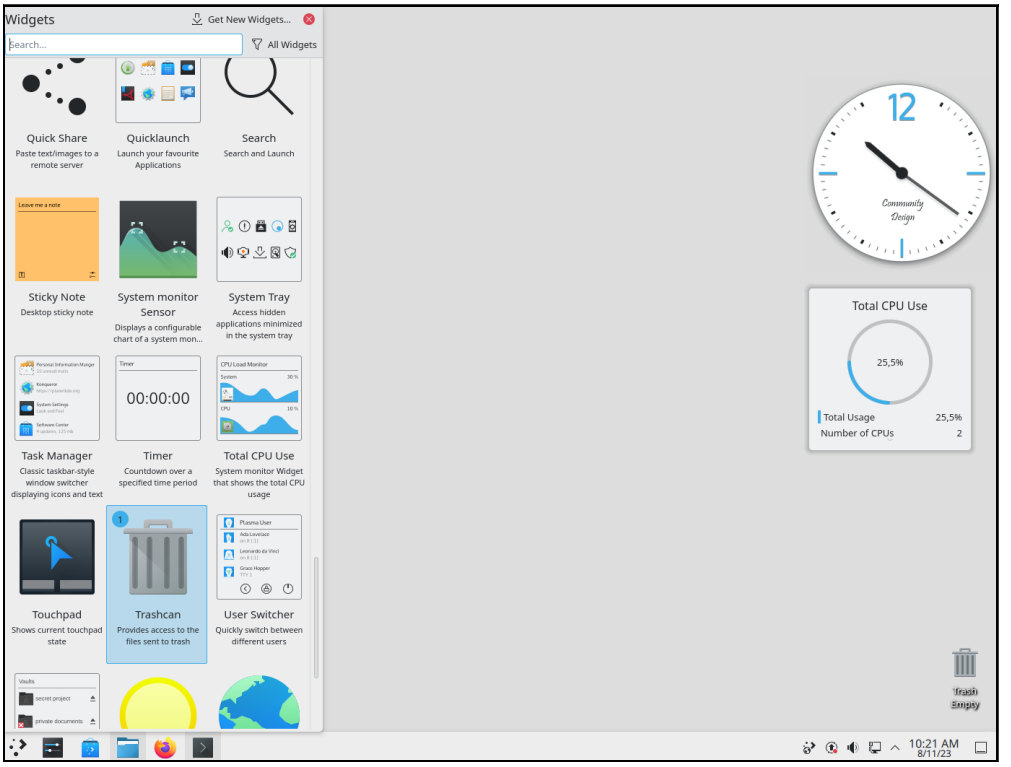


Figure 5.2 Inserting Widgets (Plasmoids)

All Plasmoids to the Front Row, Please!

Personally, I am not a fan of icons, applets, and other desktop objects. For me, several large windows usually cover the entire workspace. If you like to use icons and plasmoids, you should keep in mind the keyboard shortcut Ctrl+F12: it brings the desktop elements to the foreground and puts all windows dimmed in the background. Pressing Ctrl+F12 again or Esc restores the previous desktop state.

Once placed on the desktop, plasmoids cannot be moved with the mouse or otherwise configured. You have to use the **Start Edit Mode** context menu to switch to a special mode where you can then move, enlarge, reduce, and, if necessary, delete the programs.

Panels

The panel or (in KDE terminology) the *control bar* is located at one edge of the screen (at the bottom by default). The panel itself has no function, but only serves as a container for widgets. Even such basic elements as the menu and the taskbar are implemented as plasmoids in KDE. For this reason, it is basically possible (although uncommon) to do without a panel altogether and place the menu, taskbar, and other typical panel content directly on the desktop. The main advantage of the panel is that this area cannot be covered by windows. In addition, the compact arrangement of multiple plasmoids in one panel saves a lot of space.

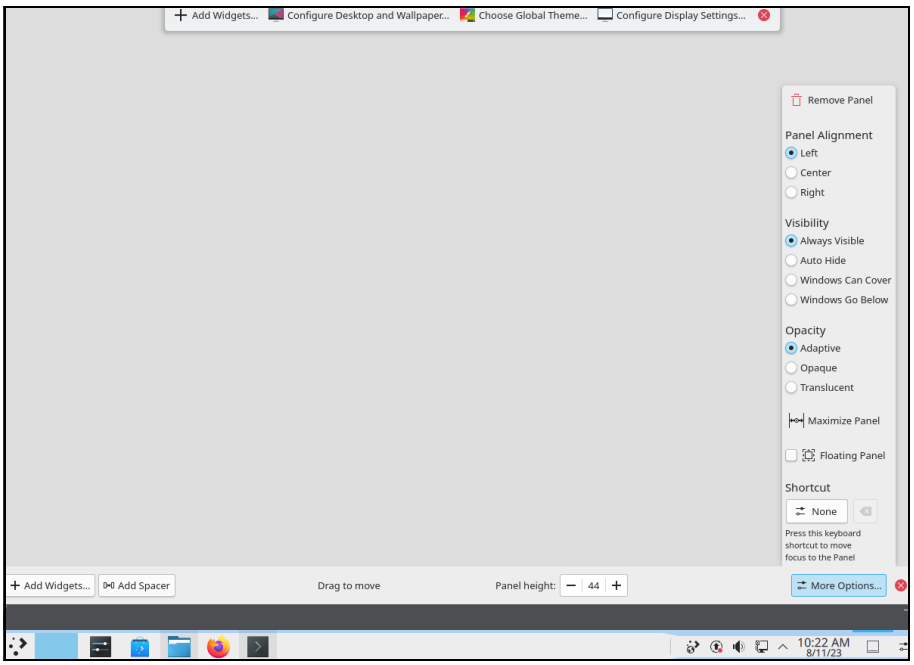


Figure 5.3 Panel Configuration

A **Settings** button on the edge of the panel allows you to change the size, position, and other properties of the panel, as well as add, move, and remove plasmoids (see Figure 5.3). For this purpose, a kind of menu is displayed above or next to the panel. By the way, the color or background graphic of the panel is predefined by the desktop theme and can only be changed by selecting a different theme (Section 5.4).

5.2.1 Important Plasmoids

KDE menu

Probably the most important plasmoid is the application starter (bottom left in Figure 5.1). It is responsible for the KDE menu, which is divided into two categories: **Applications** displays programs sorted by different categories, and **Places** helps you to quickly open important directories. In addition, the menu contains buttons to shut down or restart the computer.

One important part of the KDE menu is a search function, which is particularly suitable for launching programs quickly without navigating through menus.

You can drag and drop frequently used programs to an empty area of the panel or desktop. They appear there as icons and thus enable a particularly fast start.

Taskbar (window bar)

The Window Bar plasmoid displays an icon for each window and thus functions like the Windows taskbar. Several windows of a program are automatically combined into a group. Programs that you use frequently can be pinned to the window bar and subsequently launched in a particularly convenient way.

Workspaces

Workspaces allow you to distribute the windows of running programs to several virtual desktops and to switch between them. This makes work easier and improves the overview when you open a lot of windows at the same time. The Workspace Switcher plasmoid is responsible for managing the workspaces. In its **Settings** menu, you can set the desired number of workspaces as well as various other options.

Setting the Number of Workspaces

Some distributions have only one desktop by default. The workspace switcher then disappears from the panel and the option to set the number of workspaces is also missing.

To change this, in the **Workspace Behavior • Virtual Workspaces** module of the system settings, you must define the number and names of the workspaces.

System section

In the system section of the panel, background programs can display an icon to draw attention to themselves—for example, when new updates are available, when the network connection changes, or when new email has arrived. The system section is usually located on the right edge of the panel and does not perform any function by itself, but only serves as a placeholder for status icons. This feature seems to be self-evident, and in fact you will probably not become aware of the system section until it is missing from the panel for some reason, and notifications about emails, updates, and the like fail to appear.

5.3 File Manager

Views

The *Dolphin* program is the file manager of KDE (see Figure 5.4). Its basic functions are quickly explained. The files are displayed in the center of the window, and there are three display modes: icons, details, and compact, which you can activate using Ctrl+1 through Ctrl+3.

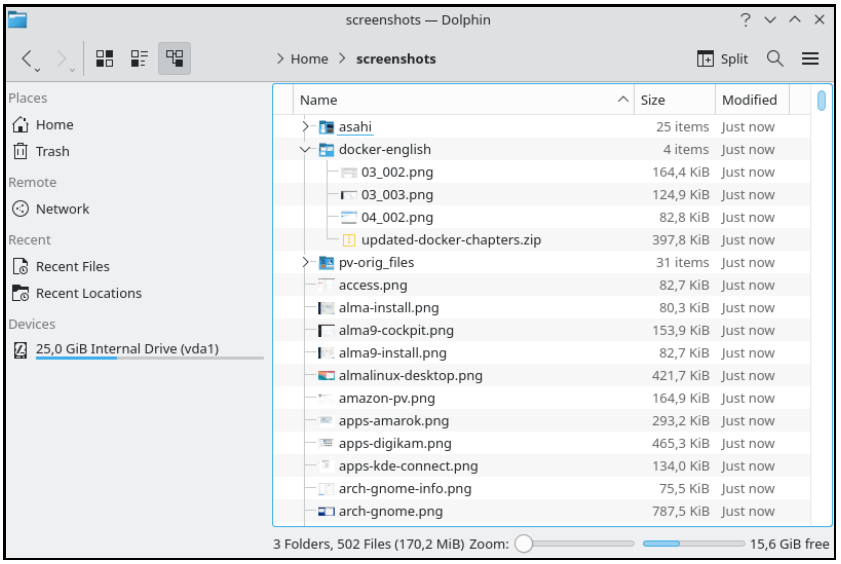


Figure 5.4 Detail View of Dolphin

The grouping function is particularly useful: **More • View • Show in Groups** groups files of the same type or by initial letter.

Using the **Preview** button, you can activate a preview for images and documents regardless of the display mode. You can adjust the size of the preview images in icon mode by using a slider. By default, Dolphin does not create a preview if the file is located in a network directory. You can control

the preview behavior via **Configure** • **Configure Dolphin** in the **General** • **Previews** dialog sheet.

For move and copy operations, the inner area can be divided vertically using **Split** to display two directories in the same window.

Directory path The current directory is displayed in a navigation bar below the menu. Ctrl+L enables you to toggle between two types of view for this bar: either the individual directories are displayed as buttons, which allows for a quick directory change, or the directory is displayed in text form, which allows a quick entry of another directory.

Sidebars On the left, the right, and below the actual window content you can display a terminal, the directory hierarchy, a list of frequently used places, and additional information via **Show Panels** or using the F4, F7, F9, and F11 keys. You can add bookmarks for directories to the list of places by dragging and dropping.

Menu Depending on the configuration, Dolphin is displayed without a menu bar. The seemingly missing commands are still available via the menu button on the right edge of the window. The conventional menu bar can be switched on and off using Ctrl+M if required.

Marking files A special feature concerns the marking of files: In the KDE default setting, a simple mouse click is not suitable for this because it displays or executes the file. For this reason, you must press Ctrl (for multiple marks) or Shift (for area marks) at the same time.

Another marking mode is even more elegant: If you *hover* the mouse pointer over a file or directory for a moment, a plus sign is displayed. Clicking this icon will then mark the file. For files that are already marked, a minus sign appears, which you can use to unmark them.

Deleting files When you delete files and directories, they first end up in the trash can. To view the contents of the trash can, you need to click the corresponding entry in the **Places** sidebar (F9). Not until you select all objects there and press Del will the files be finally deleted. To immediately delete files irrevocably, press Shift+Del.

In the Dolphin settings, you can limit the maximum amount of storage used by the trash can or have files in the trash can permanently deleted after a certain period of time.

Hidden files On Linux, all files and directories whose names begin with a dot are considered hidden. Usually, Dolphin does not display these files unless you run **Show Hidden Files** in the Tools menu. You can switch the display of hidden files on and off even faster using Alt+.

Access rights

To ensure that not every user can read or change all files and directories, Linux stores the owner and access rights for every file and directory. The underlying concept is described in detail in Chapter 9, Section 9.6. To change the owner or permissions, right-click the file and select **Properties • Permissions**.

5.3.1 Renaming Files (KRename)

Dolphin can rename multiple files, but it provides little flexibility in doing so. The relatively old *KRename* program (see Figure 5.5) offers a solution: it allows you to efficiently rename files, add a sequential number, include the date in the file name, and so on.

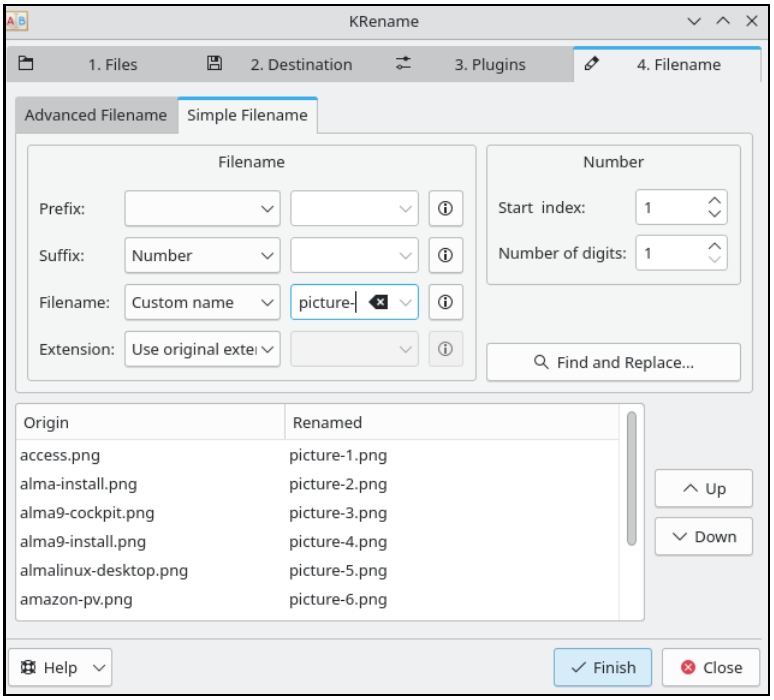


Figure 5.5 Renaming Files Using KRename

Even regular expressions are allowed for the purpose of renaming. Before the files are actually changed, a preview shows the old and new file names. KRename is an indispensable tool if you want to reorganize a large collection of audio or image files! Many distributions require you to install the program before using it for the first time. The package name is usually `krename`.

5.3.2 External Media and Network Directories

Access to removable storage media

The **Places** sidebar (F9) contains, among other things, a list of all hard disk partitions that can be mounted in the file system at a mouse click. When you connect a USB flash drive, a corresponding note appears in the panel. A mouse click then opens the file manager and displays the contents of the disk.

Before you disconnect the USB media from the computer, you must execute the **Safely Remove** command from the context menu in Dolphin in the **Places** sidebar. Alternatively, you can also find this function in the **Disk and Devices** icon in the panel.

Access to network directories

You can access the local network via the **Places** sidebar or by specifying the address, `smb://`. To access a specific directory on a Samba or Windows server directly, you want to use the following notation: `smb://servername/sharename`. This notation is also necessary if Dolphin does not detect any Windows servers on the network, which often happens depending on the firewall and network configuration.

> **Note**
>
> If Dolphin cannot find any Windows or Samba servers on the local network, your distribution's firewall may be the cause. In both Fedora and SUSE, the default firewall settings prevent the use of Windows network directories. This can be resolved by configuring the firewall correctly.

Password management

When connecting to the Windows computer or Samba server, Dolphin asks for the user name and password. This data is stored by *KDE Wallet*, a key-management service. Other KDE programs (such as mail clients) also store their login data there.

SSH

Dolphin also allows you to communicate with another computer and to copy files using the secure SSH protocol. To do this, you need to enter the address in the following format: `fish://username@computername/`. After login, Dolphin shows all files of the external computer.

5.4 KDE Configuration

The **System Settings** area contains countless configuration modules (see Figure 5.6). Because it isn't always easy to find the right module, the KDE developers have equipped the program with a search function in which you can search for keywords (e.g., "window").

The numerous modules of the system settings are presented either in an icon view or in a sidebar. To switch between the two display formats, you can use the **Setup** button.

KDE also contains configuration modules that do not affect the desktop but do affect system settings—for example, for network configuration or the printer. These modules are very useful for distributions that do not provide their own configuration tools. On SUSE, however, you should prefer YaST for configuration purposes. You may have to switch to an admin mode first by specifying the `root` password for configuration modules that affect system settings.

When operating the modules, note that changed settings do not become effective until they are confirmed via the **Apply** button.

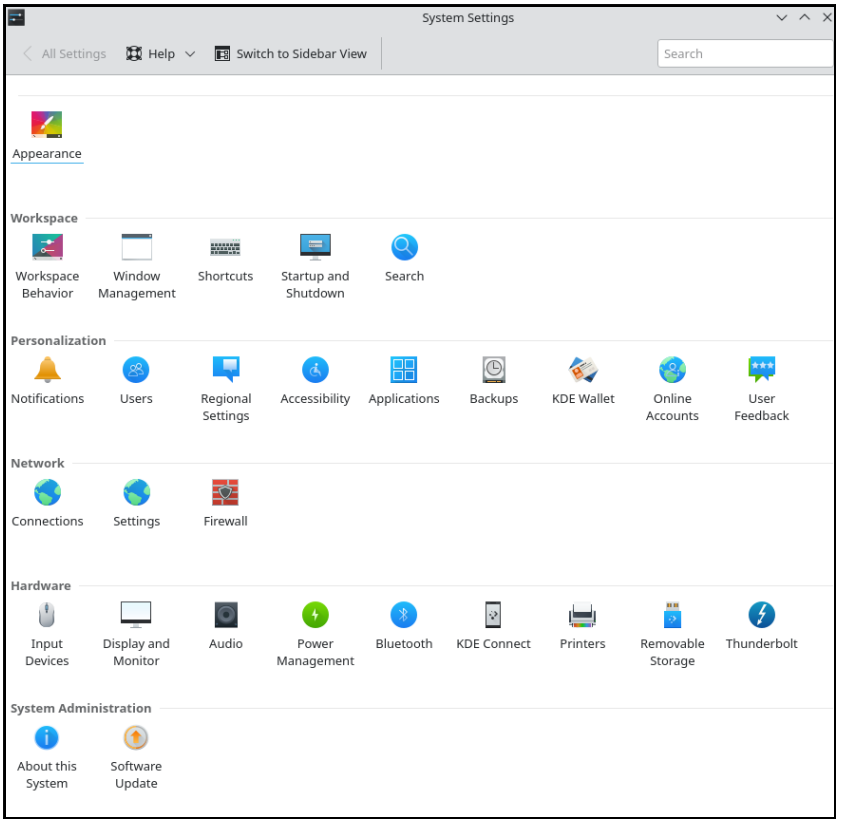


Figure 5.6 KDE System Settings in Icon View (Active by Default in openSUSE)

Configuration directories

Many KDE programs conform to the XDG standard and use the XDG directories already listed in Chapter 4, Section 4.9:

`~/.cache/`	(cache)
`~/.config/progname/`	(configuration settings)
`~/.local/share/progname/`	(user data)

Some KDE programs store their settings in files of the `.kde` directory instead. Among others, the following subdirectories are located there:

`~/.kde/Autostart/`	(personal autostart programs)
`~/.kde/share/config/`	(configuration settings)
`~/.kde/share/apps/`	(other program-specific files)

Auto login

After starting the computer, you need to log in before you can start working. If you are the only user of the computer and there is no risk of other people having access to the computer, you can automate the first login at computer startup. You can control the autologin function in the **Startup and Shutdown • Login Screen** dialog sheet. Behind the scenes, the KDE display manager (`sddm`) is usually responsible for the login.

In SUSE distributions, the autologin feature is configured independently of the desktop in the `/etc/sysconfig/displaymanager` file. Do *not* try to change the autologin via KDE tools: your settings will be overwritten by YaST at the next opportunity!

Autostart

Every time you log out, all running programs are terminated. The next time you log in, KDE will try to restart the programs that were running last—that is, restore the last session. For KDE programs, this mostly works well; for all other programs, it works only with restrictions or not at all. You can set details about this behavior in the **Startup and Shutdown • Workspace Session** module in the system settings.

Independently of the session management, you can specify programs in the `.kde/Autostart` directory that should be started after each login. KDE expects `*.desktop` files in this directory that describe the program to be launched. The easiest way to create such files is to open the `.kde/Autostart` directory via the Dolphin file manager and copy the relevant program from the KDE menu to it via drag and drop. Alternatively, you can use the **Startup and Shutdown • Autostart** control module for configuration.

KDE considers the following autostart directories:

`~/.kde/Autostart/`	(personal autostart programs)
`/usr/share/autostart/`	(global autostart programs for KDE)
`/etc/xdg/autostart/`	(global autostart programs for GNOME and KDE)

Monitor configuration

The **Display and Monitor** module enables you to define whether and how several monitors or signal outputs are to be used and in which resolution

you want to work. For high-resolution monitors, the **Scale Display** button opens another dialog where you can freely set a scaling factor.

Desktop appearance

There are numerous ways to influence the appearance (look) of the desktop. If you have the time and inclination, you can spend hours customizing the desktop according to your own ideas:

- **Desktop background**
  To set the background, right-click the desktop and select **Configure Desktop and Wallpaper**. Then set a background image or color.
- **Desktop design**
  In the **Appearance • Global Design** module, you can set the theme for the desktop. This determines the basic settings for the appearance of the panel, KDE menu, window decoration, and so on, as well as the colors used for them.

  The **Download New Global Designs** button allows you to download more designs from the *https://store.kde.org* website. Don't forget to activate the new design by applying it!
- **Controls**
  In the **Application Style** system settings module, you can choose from multiple layout variants for the visual design of buttons, radio buttons, scroll bars, and so on.
- **Panel and other desktop elements**
  In the **Plasma Style** module, you can choose from several layout variants that change the appearance of the panel.
- **Windows**
  In the **Window Decorations** module, you can choose from multiple window layouts. These enable you to change the color and design of the window frame. In the **Titlebar Buttons** dialog sheet, you can also set which buttons should be placed where in the window bar.
- **Colors**
  The colors for the window frames, the menu, the panel, and so on are basically predetermined by the **Global Design** settings and the window decoration variant. The **Color** module allows you to select other color schemes.

Figure 5.7 shows an example of how much you can change the appearance of the KDE desktop with just a few options:

- The panel was placed on the left edge of the screen.
- The panel elements have been reduced to the minimum.
- The number of window buttons has been reduced.

- The bold font makes the window title easier to read.
- The desktop was cleared of all plasmoids.

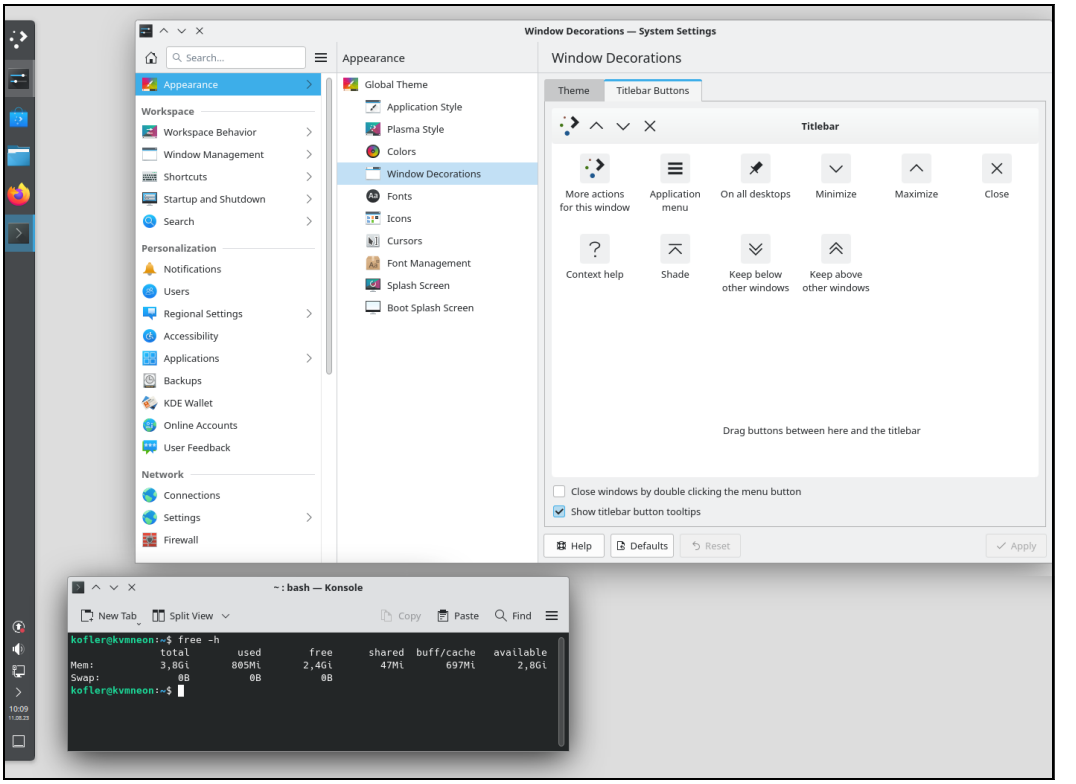


Figure 5.7 Simplified KDE Configuration

Keyboard and mouse

You can set the keyboard layout in the **Input Devices** module. In the **Keyboard • Layouts** dialog sheet, you can set up multiple keyboard layouts for quick switching. You can also set countless options there that control, for example, what functions the Windows, Alt, and Ctrl keys have.

The options for configuring the mouse or touchpad behavior can also be found in the **Input Devices** module of the system settings. There you can not only set the double-click behavior, but also specify the desired scroll direction: normal or reverse, as on smartphones or on macOS.

Printer configuration

The KDE Control Center can also be used to configure printers via the **Printers** module. You can launch the wizard for printer configuration using **Add Printer**. As a rule, the module automatically detects printers connected by cable or accessible on the network. After selecting the printer, you often have a choice of multiple drivers.

MIME

If an audio player appears on the screen after clicking an MP3 file, then KDE's MIME settings are responsible for this. MIME stands for *Multipurpose Internet Mail Extensions* and is a database that creates a mapping between file types and programs. You can change the list of MIME file types in the **Applications • File Associations** dialog sheet of the system settings. Individual file types can be assigned to multiple programs. The highest ranked program will be launched when the file is opened by a mouse click. All other programs are available for selection when you right-click and select **Open With**.

Setting default programs

By default, KDE mostly uses Konqueror as the web browser, KMail or Kontact as the email program, and `konsole` as the console program. If you want KDE to launch other programs when you click the corresponding links, you will find the setting options in the **Applications** module of the system settings.

PART III

Linux Basics

Chapter 6
Using the Terminal

From desktop to terminal

Up to this point, I have presented Linux to you primarily as a desktop system. You have become acquainted with various programs that may look a little different from Windows or macOS, but ultimately serve the same purpose and are similar to use. However, using Linux does not end at this point. There is another side to Linux that may seem daunting at first glance. Experienced Linux users execute commands in terminal windows or text consoles and receive the results also in text format. The mouse only plays a minor role; graphical user interfaces are a thing of the past.

Once you're familiar with working in terminal windows, you can perform many tasks efficiently there. You can link Linux commands with each other, execute them in the background, automate them in small programs (scripts), and so on. All of these options are available to you even if you are not sitting at the computer locally, but only have a network connection.

Office users will find less use for terminal windows than programmers or network administrators. In any case, working in the terminal belongs to the elementary tools of the trade of every user who wants to get to know Linux properly. You'll notice this for the first time at the latest when the graphics system doesn't work due to a misconfiguration or when you want to administrate your external root server.

Cross references

This chapter only provides a first overview of working techniques in terminal windows or consoles. A shell is responsible for the execution of the programs in the terminal. On Linux, there are multiple shells to choose from. The most commonly used is `bash`, whose basic functions are the subject of the next chapter. Alternatively, `zsh` addresses advanced users (see Chapter 8).

The other chapters then introduce various Linux commands in more detail. These are used, for example, to manage the file system (`ls`, `cp`, `mv`, `ln`, `rm`, etc.), to search for files (`find`, `grep`, `locate`), to control network functions (`ping`, `ip`, `ssh`), and so forth. Along the way, you'll learn a lot of Linux basics.

6.1 Text Consoles and Terminal Windows

Text consoles

You use Microsoft Windows exclusively in graphics mode. Linux, on the other hand, can also be used via text consoles. Most distributions have six

text consoles available. If the text mode is already active, press Alt+F1 to switch to the first console, Alt+F2 to switch to the second console, and so on. Alt+← or Alt+→ jumps to the previous or next console.

On the other hand, if the computer is running in graphics mode, Ctrl+Alt+F# will take you to the #th text console. However, there is no uniformity among the distributions about which console is intended for which task. In many distributions, the first console runs the display manager with the login box, while the second console runs the desktop system. This blocks the first two consoles, and Ctrl+Alt+F3 takes you to the first text console that is available.

You can return to the desktop system using Alt+F2 in most distributions, but depending on the distribution, Alt+F1 or Alt+F7 can also take you there. Just try it out!

Before you can work in a text console, you must log in. When you have finished working or if you wish to log in under a different name, you must log out again. To do this, simply press Ctrl+D.

Shortcut	Function
Ctrl+Alt+F#	Switch from graphics mode to text console #
Alt+F#	Switch from one text console to text console #
Alt+F2	Return to graphics mode (depending on the distribution, also Alt+F1 or Alt+F7)
Alt+←/+→	Switch to the previous/next text console
Shift+Page ↑/Page ↓	Scroll forward/backward through the text outputs of the console
Ctrl+Alt+Del	Shutdown Linux (only in text consoles; executes `shutdown`, so be careful!)

Table 6.1 Keyboard Shortcuts to Activate Text Consoles

You can start a command in one console, and while this command is running, you can do something else in the second console. You can also log into one console as `root` to perform admin tasks while editing a file in the other console under your normal login name. This means that each console runs completely independently of the others.

Using Shift+Page ↑ and Shift+Page ↓, you can scroll up and down in the screen contents of a text console. In this way, you can view the results

of the most recently executed programs again, even if they have already been moved out of the visible screen area.

6.1.1 Terminal Window

Usually you work in text consoles only when there is no other option—for example, on a server where no desktop system is available at all, or in case of configuration problems when the graphics system cannot be started.

The normal case, on the other hand, is that you first log into your desktop system and then open a *terminal window* to execute commands (see Figure 6.1). Depending on the distribution and desktop system, different terminal programs are available for selection. The most popular ones include `gnome-terminal`, `kgx` (both GNOME), and `konsole` (KDE). The menu command for starting a terminal window also varies depending on the distribution. Here are two examples:

- In distributions with GNOME, press Windows, type `terminal`, and then press Enter
- In distributions with KDE, flow menu path **Applications • System • Terminal**

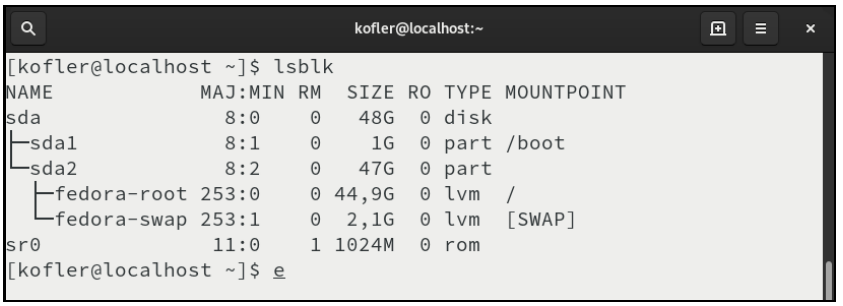


Figure 6.1 Terminal Window

In terminal windows, you can basically work like you can in a text console. The biggest advantage is that you can scroll through previous issues more conveniently thanks to a scroll bar.

Important keyboard shortcuts

Within text consoles or terminal windows, various keyboard shortcuts help you to enter commands efficiently. Table 6.2 summarizes the most important shortcuts. They only apply if you use `bash` as a shell in the default configuration, which is the case for most distributions. If you work in a terminal window on GNOME, you should run **Preferences / General** and uncheck the **Enable mnemonics** option.

Shortcut	Function
Ctrl+A	Places the cursor at the beginning of the line (like Home)
Ctrl+C	Cancels the program
Ctrl+E	Places the cursor at the end of the line
Ctrl+K	Deletes the line to the right of the cursor
Ctrl+Y	Reinserts the last deleted text
Ctrl+Z	Interrupts the program (continue via fg or bg)
Tab	Completes file and command names
↑/↓	Scrolls through the commands executed so far

Table 6.2 Keyboard Shortcuts for Entering Commands in bash

Command completion via Tab especially saves a lot of typing work. You only need to specify the first letters of a command or file, then press Tab. If the command, directory, or file name is already clearly recognizable, it will be completed in full; otherwise only until several options arise. Pressing Tab twice will cause a list of all file names beginning with the initial letters already entered to be displayed. This feature is described in detail in Chapter 7, Section 7.3.1.

Showing and Hiding the Menu in gnome-terminal

In gnome-terminal, you can hide the menu via **View • Menu Bar**. Often the program is configured this way by default. But how do you show the menu again? To do this, right-click the terminal or press Shift+F10. In the context menu that opens, select the **Show Menu Bar** command.

Clipboard

Because a lot of keyboard shortcuts with Ctrl have a preassigned meaning in the terminal, you have to think differently when copying and pasting text: To copy a text segment previously selected with the mouse to the clipboard, you need to press Shift+Ctrl+C. To insert text again, press Shift+Ctrl+V accordingly.

Some other keyboard shortcuts depend on the terminal program. For this reason, some of the shortcuts summarized in Table 6.3 are valid only in the gnome-terminal program.

Shortcut	Function
Shift+Ctrl+C	Copies selected text to the clipboard
Shift+Ctrl+N	Opens a new terminal window
Shift+Ctrl+T	Opens a new dialog sheet (tab)
Shift+Ctrl+V	Pastes text from the clipboard
Ctrl+Page ↑	Switches to the previous dialog sheet
Ctrl+Page ↓	Switches to the next dialog sheet
Ctrl++/+-	Changes the font size

Table 6.3 Keyboard Shortcuts to Control Terminal Program (Partly GNOME-Specific)

Mouse

The mouse plays only a minor role in text consoles or terminal windows. You *cannot* use it to change the current cursor position! This can only be done using the cursor keys. The function of the mouse is limited to copying text using the left mouse button and then pasting it back in using the middle mouse button or by pressing the mouse wheel at the current cursor position. This is incredibly useful and efficient.

If you have a mouse without a mouse wheel or a touchpad without a middle button, you must first copy the text to the clipboard via a keyboard shortcut and paste it back in using a second shortcut, as in other operating systems. It couldn't be more inconvenient! Now you may understand why many Linux geeks look for the touchpad to have a third button when buying a notebook computer.

The Mouse in Text Consoles

In text consoles—that is, when you do *not* work in a graphical user interface—you can't actually use a mouse unless you install and run the `gpm` program. After that, you can also use the mouse in text consoles to copy text.

Colors

Most terminal programs are preconfigured so that the background is dark and the font is light. Of course, you can change this in the settings of your terminal program. However, you should note that some commands executed in the terminal use colors and rely on the background being dark. If you now set an eye-friendly light background, some output may be difficult to read.

6.1.2 Running Commands

To execute commands, you simply need to type the command name in the text console or terminal window, possibly some parameters, and finally press Enter. The `ls` command returns a list of files and subdirectories in the current directory:

```
user$ ls -l
-rw-------  1 user users   506614 29. Jun 12:11 offer-katzbauer.pdf
drwxrwxr-x  3 user users     4096 13. Apr 11:31 bak
drwxrwxr-x  2 user users     4096 18. Jul 15:03 bin
-rw-r--r--  1 user users   243571  3. Jul 09:14 DB20078.jpg
drwxr-xr-x  2 user users     4096  7. Apr 10:59 Desktop
-rw-------  1 user users 17708403 May 19 10:35 DN.pdf
...
```

The preceding example shows how the command input and result are represented in this book: `user$` at the beginning of the first line means that the command was executed by an ordinary user named `user`. If `root#` is specified in the first text column instead, the command was executed by `root` (i.e., by the system administrator). Either `user$` or `root#` is considered the input prompt. These characters are automatically displayed at the beginning of each input line. You must *not* enter these characters yourself!

In general, only the characters in bold type in this book must be entered! On your computer, instead of `user$` or `root#`, you may see some other text, often containing the current directory and/or computer name. For reasons of clarity, I have omitted this information in this book.

Sometimes there isn't enough space in this book to print a command on a single line. In such cases, the command is spread across multiple lines separated by the \ character, which may then look as follows:

```
user$ gconftool-2 \
  --set "/apps/panel/toplevels/top_panel_screen0/monitor" \
  --type integer "0"
```

You can now enter this command on two lines, but if you do, then you must end the first line with \ as in the book. However, you can also simply drag the two lines together: in that case, the \ character must be omitted.

Running commands in the background

You can also execute commands in the background. This means that you do not need to wait for the program to end, but can continue working immediately. To do this, enter the `&` character at the end of the command line. This procedure is especially recommended if you start a program with a graphical user interface from a console (e.g., `firefox &`).

Working as root

It's unusual on Linux to work as `root`—that is, with system admin rights. If you are logged in as an ordinary user, there are several ways to execute individual commands with `root` privileges. For most distributions, you simply run `sudo -s` in the terminal window. After entering your password, you will have `root` privileges. Then you can execute the relevant commands. Either `exit` or Ctrl+D returns you to the original user. Tips on how to move the command execution from the foreground to the background, how to determine a list of all active commands (processes), and so on follow in Chapter 10. I discuss `sudo` in detail in Chapter 10, Section 10.3.

6.2 Displaying and Editing Text Files

The less command

You can of course open text files in an editor in KDE or GNOME.

On the other hand, if you are working in a text console or terminal window, it is best to use the `less` command to view files.

You can also place the command after other commands to read their often very long outputs page by page at your leisure or to search for text in them (see Table 6.4):

```
user$ less datei        (page-by-page display of the file)
user$ ls -l | less      (page-by-page display of the file directory)
user$ ps ax | less      (page-by-page display of the list of processes)
```

On some mini-Linux systems—for example, in embedded devices like NAS hard disks—the `less` command is not available. The `less` predecessor, `more`, might be installed instead. Otherwise, you can use `cat` to output the entire contents of a text file—that is, without scrolling page by page. If you want to read only the last lines, such as in a logging file, then you can use `tail`.

The Terminal Shows Only Cryptic Characters...

If you display a file in a text console that contains binary data instead of text, the data may be interpreted as special characters and the console may be confused. In this case, only cryptic characters are displayed on the screen or in the terminal window because the assignment of the character set is no longer correct. You can usually solve this issue by using the `reset` command.

Shortcut	Function
Cursor keys	Move text up or down
Home, End	Jumps to the beginning/end of the text
G, Shift+G	Jumps to the beginning/end of the text
/ [text] Enter	Forward search
? [text] Enter	Backward search
N	Repeats the forward search (*next*)
Shift+N	Repeats the backward search
Q	*Quit*
H	Displays help text with additional keyboard shortcuts

Table 6.4 Keyboard Shortcuts for less

6.2.1 Text Editors

On KDE or GNOME, `kate`, `gedit`, or `gnome-text-editor` are convenient text editors with intuitive operation. However, these programs cannot be used in a text console; you need an editor that runs in text mode. This section presents the most popular representatives of this category. Depending on your distribution, you may have one of the editors installed by default.

The nano editor

The `nano` editor has a modest range of commands but is easy to use. The two bottom lines on the screen provide an overview of the available commands (see Figure 6.2).

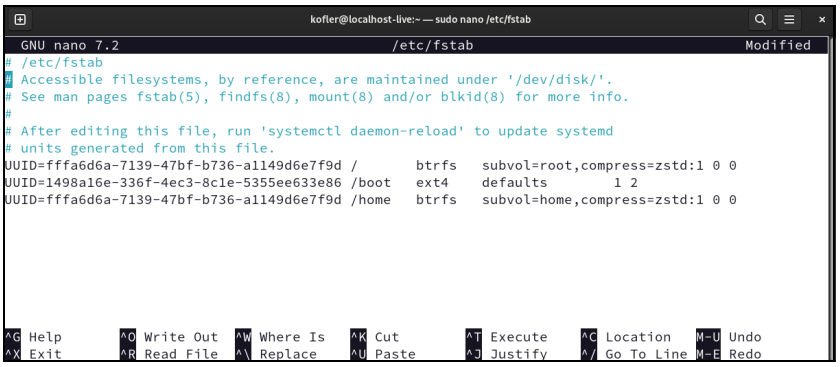


Figure 6.2 nano Editor in Terminal Window

Emacs, jove, jed, jmacs

GNU Emacs plays a special role among editors. It's hard to imagine the everyday life of many administrators without this program. I'll introduce

this editor in more detail in Chapter 13. For now, Table 6.5 summarizes the most important keyboard shortcuts. They also apply to the `jove`, `jed`, and `jmacs` editors. These are minimized versions of Emacs, which are compatible in their basic functions.

Shortcut	Function
Ctrl+X, Ctrl+F	Loads a new file
Ctrl+X, Ctrl+S	Saves the current file
Ctrl+X, Ctrl+W	Saves the file under a new name
Ctrl+G	Cancels the input of a command
Ctrl+K	Deletes a line
Ctrl+X, U	Undoes the deletion
Ctrl+X, Ctrl+C	Exits Emacs (with prompt to save)

Table 6.5 Emacs Keyboard Shortcuts

Vi, Vim, and Elvis

Another veteran of Unix history is the Vi editor, which is usually represented in Linux by the compatible Vim program, less frequently by the equally compatible Elvis editor. The original Vi is not part of Linux for copyright reasons. However, the `vi` command can still be executed and then causes Vim or Elvis to launch.

Vi offers almost as many functions as Emacs, but its operation is even more difficult to learn. In return, Vi is comparatively compact and is usually also available on emergency systems. The program represents an unofficial Unix/Linux standard and is automatically called as an editor by various programs.

Its main fundamental difference from other editors is that Vi distinguishes between different modes. Text input is only possible in insert mode (see Table 6.6).

Shortcut	Function
I	Switches to insert mode
Esc	Exits the insert mode
H/L	Moves the cursor to the left/right
J/K	Moves the cursor down/up
X	Deletes a character

Table 6.6 Vi Shortcuts

Shortcut	Function
D + D	Deletes the current line
P	Reinserts the deleted line at the cursor position
U	Undoes the most recent change
:	Switches to the complex command mode

Table 6.6 Vi Shortcuts (Cont.)

Most commands are entered in complex command mode, which can be activated using : (see Table 6.7). If necessary, the insert mode must be exited by pressing Esc prior to that. The cursor movement is of course also possible by using the cursor keys. Chapter 12 is dedicated to Vi in its Vim manifestation.

Shortcut	Function
:w *name*	Saves the text under a new name
:wq	Saves and exits Vi
:q!	Exits Vi without saving
:help	Starts the online help

Table 6.7 Vi Commands in Complex Command Mode

joe

The joe editor is simple. Its keyboard shortcuts are modeled after the WordStar word processor from DOS times. To obtain a full description of all its commands, you can run man joe in a console. Ctrl+K, H displays a help text with the most important keyboard shortcuts.

The program can also be started under the name jmacs or jpico. Then other keyboard shortcuts apply, which are compatible with Emacs or nano. Personally, I'm particularly fond of jmacs. This is a minimalistic Emacs clone that is sufficient for most purposes. That's why joe is often the first package I install on a new server.

Setting the default editor

Some programs automatically start an editor to view or edit files, and that's usually the Vi editor by default. If you prefer a different editor, you must set the EDITOR and VISUAL environment variables in /etc/profile or .profile:

```
# addition in /etc/profile or in ~/.profile
export EDITOR=/usr/bin/jmacs
export VISUAL=$EDITOR
```

In addition, many distributions provide the option to set a default program for several functionally similar programs (editors, Java interpreters, etc.) via the `alternatives` command; see Chapter 16, Section 16.10.

6.3 man and info

Commands like `ls`, `cp`, or `top`, which you usually run in a terminal window, do not respond to F1 nor do they have a **Help** menu. But there are also help texts for these commands available, which can be read via other commands:

- `command --help` returns a list of all options with a short explanation of their meaning for a large number of commands.
- `man command` displays the `man` help text for many commands. You can navigate through the multipage text using the cursor keys. Press Q to exit the help.
- `help command` only works with shell commands, such as `cd` or `alias`.
- `info command` is an alternative to `man`. The `info` system is especially suitable for very extensive help texts. Whether the help text is in the `man` or `info` system simply depends on which help system the program developers have chosen—`man`, however, is clearly more popular.

man

`man` is a command to display the documentation of many elementary commands like `ls` or `cp`. `man` can be called as `man command` to read the help text for `command`.

The optional specification of a range (`man range command`) restricts the search for `man` texts to a range of topics. For example, `man 3 printf` returns the syntax of the `printf` C function. This restriction is necessary if several `man` texts with the same name exist in different subject areas. in this case, only the first `man` text found will be displayed.

If you want to read all `man` texts with the same name from all areas, you must use `man` with the `-a` option. As soon as you have read the text and exited `man` by pressing Q, the `man` text for the subsequent section will display.

In many Unix and Linux books, the `man` numbers are given together with the commands—for example, `find(1)`. This way you know immediately how to call `man`. `man` usually knows topic ranges 1 through 9 and n (see Table 6.8). Sometimes the commands of programming languages are classified in additional areas using other letters.

	Topic		Topic
1	User commands	6	Games
2	System calls	7	Miscellaneous
3	C functions	8	System administration commands
4	File formats, device files	9	Kernel functions
5	Configuration files	n	New commands

Table 6.8 man Topic Groups

The help texts are displayed internally by the `less` program. For this reason, the keyboard shortcuts from Table 6.4 are valid for the navigation in the help text. The directories from which the help texts are read by `man` can be set either via the `/etc/manpath.config` control file or via the `MANPATH` environment variable.

On KDE and GNOME, you can also read `man` texts using the respective help or web browsers. The following examples show how you can display the `man` page for `ls` and a table of contents of all `man` pages:

```
user$ gnome-help man:ls
user$ khelpcenter man:ls
user$ khelpcenter 'man:(index)'
```

help For some commands, you can get help not via `man` but via `help`. This applies to all commands that are executed directly by the shell. (The shell is the command interpreter that takes your input. Detailed information about the Linux standard shell `bash` follows in the next chapter).

info `man` help texts have the disadvantage that they are difficult to structure. The entire text must be reproduced in a single—sometimes almost endless—document. The alternative `info` format provides much better options here, which is why extensive help texts in particular are often only available in this format. `info` is usually called in the format `info commandname` (such as `info ls`) and is controlled via various keyboard shortcuts (see Table 6.9). If the command is started without parameters, the program displays an overview of the available help topics.

Shortcut	Function
Space	Scrolls text down
Backspace	Scrolls text up

Table 6.9 info Keyboard Shortcuts

Shortcut	Function
B, E	Jumps to the *beginning/end* of the info unit
Tab	Moves the cursor to the next cross-reference
Enter	Traces the cross-reference to another info unit
N	Next info unit of the same hierarchy level (*next*)
P	Previous info unit of the same hierarchy level (*previous*)
U	One hierarchy level up (*up*)
L	Back to the last displayed text (*last*)
H	Detailed user manual (*help*)
?	Command overview
Q	Exits `info` (*quit*)

Table 6.9 info Keyboard Shortcuts (Cont.)

Unfortunately, the advantage of a clearer structuring quickly turns out to be a disadvantage: navigating in `info` texts is confusing, and there is no search mechanism that extends beyond the current page.

Chapter 7
Bash (Shell)

The focus of this chapter is the *Bourne Again Shell*—or `bash` for short. This program allows you to run commands in a terminal window or in a text console. A *shell* is a command interpreter that provides numerous additional functions, such as the combination of several commands or the storage of the results of a command in a file. At the same time, `bash` contains its own programming language that can be used to create shell programs (shell scripts).

This chapter describes the use of `bash` both as a command interpreter and for programming purposes. Essential topics in this chapter are an introduction to using `bash`, input and output redirection, communication between multiple processes (pipes, command substitution), and variable management. If you are interested in `bash` programming, see Section 7.8 for an overview of the main language elements and various examples. The chapter ends with a table of all special characters of `bash`.

zsh—the shell for professionals

`bash` provides numerous functions and is widely used. For Linux and shell fans, however, there is another level up: `zsh`. I will introduce `zsh` in the next chapter, but I will assume that you already know how a shell works. In this respect, it's a good idea to start with this chapter, even if you plan to switch to `zsh`.

7.1 What Is a Shell?

Bourne Again Shell is a pun: `bash` is the "reborn" Bourne shell, which used to be one of the three classic Unix shells, along with the Korn shell and the C shell. On Linux, all three shells and some others are available, but `bash` is that one that's mostly set up by default.

So, what is a shell? Primarily, the shell is used to call Linux commands and programs. It's a kind of command interpreter, comparable to `cmd.exe` or PowerShell from the Windows world. A shell is executed in every terminal window and text console after login.

At the same time, the shell provides a programming language that can be used to automate workflows. Using special shell commands, you can use variables, form queries and loops, and the like within these programs. The resulting programs are called *scripts*. Scripts are text files that are executed (interpreted) by a program—in this case, by `bash`.

Shell versus Terminal

Especially if you come from the Windows world and are familiar with `cmd.exe` or PowerShell, the distinction between the terminal program and the shell is probably foreign to you. On Linux, the terminal program is only responsible for the external user interface, if you will, for processing keyboard events and displaying text. The actual work is done by the shell that is executed *inside* the terminal.

Version

This chapter describes `bash` version 5.*n*, but it can also be applied to version 4, which can be found in older distributions. (Despite the version jump, only details have changed). If you don't know which shell (version) you're using, you should run the following commands:

```
user$ echo $0
bash
user$ bash --version
GNU bash, version 5.2.15(1)-release (x86_64-redhat-linux-gnu)
```

Documentation

There is an extensive `man` text and an equally extensive `info` file available for `bash`. You can also read the same text in a web browser at *https://gnu.org/software/bash/manual/bash.html*.

7.1.1 Other Shells

`bash` is considered by many distributions as the standard shell for working in consoles or terminal windows, but you can install countless other shells using your distribution's package management system. Among Linux professionals, the aforementioned `zsh` is especially popular (see Chapter 8). Other variants include the Korn shell (`ksh` or `pdksh`) and the C shell (`csh` or `tcsh`). To try one of these shells after installation, you need to start a terminal window and run the respective shell name in it; `exit` returns to the last active shell:

```
user$ zsh
hostname% ls     # run commands in zsh
...
hostname% exit   # back to the previous shell
user$
```

Changing the default shell

A separate default shell is provided for each Linux user. This shell is executed when you open a terminal window or when you log into a text console. The default shell is stored in the `/etc/passwd` file. To set a different default shell, you must run the `chsh` command (*change shell*):

```
user$ chsh -s $(which zsh)
```

A list of the shells that are installed in your distribution and thus usable is located in /etc/shells.

dash

Because bash is considered relatively slow and takes up a lot of memory—not least because of its many functions—some distributions use a more efficient shell to run scripts instead of bash. On Debian and Ubuntu, for example, the Debian Almquist shell (dash) is used when a script is to be executed quite generally by /bin/sh and not explicitly by /bin/bash. dash is almost but not entirely compatible with bash. If you use bash-specific functions when programming, you must explicitly specify #!/bin/bash in the first line.

7.2 Configuration

bash works right out of the box in all Linux distributions. Changes to the configuration are only necessary if you want to implement additional functions, set environment variables different from the default, and so on. You can usually make such changes in the .bashrc file. This section explains which other configuration files exist and what file is taken into account when.

A wealth of configuration files

Table 7.1 provides an overview of the most important global configuration files (first column) as well as user-specific configuration files in the respective home directory, together with information about which type of shell considers which configuration file. Note that this table is somewhat simplified and doesn't include all distribution-specific features. For example, on Debian and Ubuntu, bash reads /etc/bash.bashrc instead of /etc/bashrc. In the local directory, .profile is also analyzed if neither .bash_profile nor .bash_login exist.

Global	Local	Login Shell	Interactive Shell	Script
/etc/profile	.bash_profile	•	•	
/etc/bashrc	.bashrc	◦	◦	
/etc/inputrc	.inputrc	◦	◦	
	.bash_login	•		
	.bash_logout	•		

Table 7.1 Different bash Configuration Files Considered Depending on Type of Shell and on Linux Distribution

Consider a brief overview of the meaning of the files:

- The `profile` files are used to set environment variables. They contain information that is analyzed by the shell and by various programs (see Section 7.7.1). Environment variables can also be set in `/etc/environment`. However, this file applies system-wide to all processes and not only to `bash` or other shells.
- The `bashrc` files are used to configure `bash` functions intended for interactive use (not scripts). Custom changes to the configuration are almost always stored in `.bashrc`. So if you want to define your own shortcuts (aliases), extend the path (`PATH`) to all executable programs, or modify properties of `bash`, `.bashrc` is the right place.
- The `inputrc` files only concern the configuration of the keyboard and apply to the `readline` library, which is used internally in `bash` to process keyboard input. There is rarely a reason to touch the configuration predetermined by the distribution. The `bashrc` and `inputrc` files are *not* read by `bash` by default. However, most Linux distributions are configured to take these files into account after all.
- The `login` and `logout` files are only considered by login shells (discussed ahead) and are executed when you log in or out.

Different ways of using the shell

Table 7.1 takes into account three types of use: login shells, interactive shells, and scripts. The easiest to understand is a script: Here, `bash` is used as an interpreter to execute program code. No configuration files are loaded in the process. However, when the script is started interactively in a terminal window, the already existing settings still apply.

The difference between a login shell and an interactive shell is a bit more difficult—not least because *both* types of shells are interactive and therefore the names are not very conclusive:

- A *login shell* exists when you log in to a text console or when you work via SSH on an external computer. This means that the shell is started immediately after authentication and then (interactively) accepts and processes your commands.
- On the other hand, we speak of an *interactive shell* if the login has already happened earlier and the shell is started later—if required. This is always true if you first log into a desktop system and then open a terminal window. There again a shell is started, which waits for your commands.

Input prompt

In the shell, the name of the computer, the user, and/or the current directory is displayed at the beginning of each input line, depending on the

distribution. This string is referred to as a *prompt*. At its minimum, the prompt consists of only one character: `#` for `root` or `$` for ordinary users. Depending on the configuration, the prompt can also display the name of the computer, the name of the logged-in user, the current directory, and so on—and all this possibly in different colors.

The content of the prompt is defined by the `PS1` environment variable—on a system-wide basis often in the `/etc/bash.bashrc` file, or on Red Hat/Fedora in `/etc/bashrc`. To set the `PS1` variable individually, you need to modify the `.bashrc` file. Background information on handling environment variables is covered in Section 7.7.1. The following line causes only the current directory to be displayed as the prompt:

```
# Change in ~/.bashrc
PS1="\w \$ "
```

Within `PS1`, `\u` is a placeholder for the user name, `\h` for the host name, `\w` for the entire current directory, `\W` for the last part of the current directory, and `\$` for the prompt ending (`$` or `#`).

Color prompt

You can use `\[\e[0;nnm\]` to embed the formatting code `nn` in `PS1`. A comprehensive prompt configuration guide including a listing of all ANSI color codes can be found in the how-to document located at *https://tldp.org/HOWTO/Bash-Prompt-HOWTO*.

On my computers, I use the following setting:

```
PS1='\[\e[0;34m\]\u@\h:\W\$\[\e[0;39m\] '
```

This will display a blue prompt in the following format: `username@computer-name:directory`. However, the prompt does not show the entire path, but only the last part—such as `nautilus`, for example, if the current directory is `/usr/lib/nautilus`. This saves space if you are in a multipart directory. In addition to `PS1`, you can also set the `PROMPT_COMMAND` variable. This variable contains a command that is executed each time before `PS1` gets displayed.

More Color in Life

Colors don't play too big a role in the terminal. Relatively few commands use colors to highlight important information. A whole collection of tips on how to coax some important commands to be more colorful can be found on the following website: *https://www.baeldung.com/linux/terminal-shell-colors*.

7.3 Command Input

bash supports you with a lot of keyboard shortcuts for entering commands. In particular, you can use the ↑ and ↓ cursor keys to reedit the most recent commands you entered, which saves a lot of typing work. When you log out of a shell, the most recently entered commands are stored in the ~/.bash_history file and are thus available again after the next login.

Command lines can be modified as in a text editor; that is, you can navigate within the entered line using the cursor keys and make corrections. By default, the bash keyboard mapping adopts the most important keyboard shortcuts of the Emacs editor: Ctrl+A moves the cursor to the beginning of the line, Ctrl+E to the end, and so on. Vi fans can alternatively activate a Vi mode using the set -o vi statement in .bashrc.

Mouse and trackpad

Note that as a text-only command, the shell does not provide any mouse support. In particular, you cannot change the cursor position by mouse click. However, you can select text using the mouse and paste it at the current cursor position using the middle mouse button. For a Linux fan whose mouse or trackpad doesn't have a middle button, many terminals also provide corresponding keyboard shortcuts—usually Shift+Ctrl+C and Shift+Ctrl+V.

7.3.1 Expanding Command and File Names

By automatically expanding command and file names, bash helps you to minimize the amount of typing work required. For this purpose, you must first enter the first letters of the command or file name and then press Tab. If the name is already clearly identifiable, it will be completed in full. If there are multiple names that start with the same letters, the name will be expanded only as far as the names match. In addition, in this case you will hear a beep indicating that the file name may not be complete yet.

The easiest way to understand the file name expansion is to use an example. On my computer, the input

user$ **ema** Tab **ba** Tab

in the directory with the files for this book is automatically expanded to

user$ **emacs bash.tex**.

Here, emacs is the name of my favorite editor and bash.tex is the name of the LaTeX file of this chapter. To complete ema, bash searches all directories specified in the PATH variable for executable programs. To complete the file name, on the other hand, only the current directory is taken into account.

The expansion also works for file names that are prefixed with multiple directories. When you enter

`user$` **`ls /usr/sh`** `Tab`,

`bash` expands this input to

`user$` **`ls /usr/share/`**.

If a unique expansion isn't possible (signaled by a beep), you can simply press `Tab` again. `bash` will then display all possible additions in the lines below the current input line. On my computer, the input

`user$` **`e`** `Tab` `Tab`

results in the output of an almost endless list of all commands and programs that start with the letter `e`. The input can then be continued.

`bash` provides similar expansion functions also for the names of home directories and for variable names: `~ko` `Tab` returns `~kofler/` on my machine, for example, and when I enter `$PAT` `Tab`, I get `$PATH`.

Determining the path to a program

The automatic command expansion conceals where a program is actually located. There are several ways to find that out:

- `whereis name` searches all default directories.
- `which name` searches all directories contained in `PATH` and determines the program that would be executed if the command were entered without a path. `which` is interesting if there are several versions of a program located in different directories.
- `type name` works similar to `which`, but also takes into account commands that are built into `bash` or defined as aliases.

Program-specific expansion

When you run the `latex` command, only files ending with `*.tex` can be considered as possible parameters. With `latex na` `Tab`, `bash` should therefore only consider the `*.tex` files in the current directory. If you run `man na` `Tab`, on the other hand, only expansions for which `man` help texts exist are relevant. Similarly, there are numerous other commands and programs where the selection of possible files or parameters is limited from the outset.

When you search for suitable completions, the `complete` command can help you. Many distributions are equipped with an extensive `complete` configuration, which sometimes has to be installed separately (`bash-completion` package). The configuration is usually done by one of the following files:

```
/etc/bash_completion
/etc/bash_completion.d/*
/etc/profile.d/complete.bash
/etc/profile.d/bash_completion.sh
```

bash-complete usually works on the first try. If you also want to define your own expansion rules, you will have to familiarize yourself with the rather confusing syntax of complete. A concise description is provided by help complete and man bash (search for "Programmable Completion").

7.3.2 Important Keyboard Shortcuts

Table 7.2 summarizes the most important keyboard shortcuts of bash. The table assumes that bash is configured for emacs mode, as is the case with almost all distributions. The bash keyboard shortcuts are predefined by the aforementioned readline library, which bash uses to process input. For this reason, readline provides even more abbreviations.

Shortcut	Meaning
↑, ↓	Scrolls through the most recently entered commands
←, →	Moves the cursor backward or forward
Ctrl+A, Ctrl+E	Moves the cursor to the beginning or to the end of the line
Alt+B, Alt+F	Moves the cursor backward or forward by one word
Alt+D	Deletes a word
Ctrl+K	Deletes everything to the end of the line
Ctrl+Y	Reinserts the most recently deleted text again
Ctrl+T	Swaps the two preceding characters
Alt+T	Swaps the two preceding words
Tab	Expands the command or file name
Ctrl+L	Clears the screen
Ctrl+R	Searches for previously entered commands
Alt+.	Inserts the most recently used parameter
Ctrl+_	Undoes the last change

Table 7.2 Most Important bash Shortcuts

Inserting the most recently used parameter

The function of the Alt+. shortcut can be understood only by means of an example. Let's suppose you have just copied a file (cp name1 name2). Now you want to delete the copy in the next command. Instead of "rm name2", type "rm" and then press Alt+. bash then automatically inserts the last

command parameter used. By pressing Alt+. multiple times, you can also access all other parameters; for example, name1 can be accessed by pressing the shortcut twice.

Command search

The Ctrl+R shortcut also requires a more detailed explanation. This shortcut enables you to search for commands that have already been entered: press Ctrl+R at the start of a line and then type the first characters of the command line you are looking for. bash then automatically displays the most recently used command with these initial letters.

By pressing Ctrl+R several times, you can toggle between different suitable options. Ctrl+S works like Ctrl+R, but it runs through the list of matching commands in the opposite direction. Enter, Tab, and the cursor keys cancel the search and execute the command that has been found or allow you to edit the command line found.

Many consoles consider Ctrl+S a statement to temporarily stop the output. (This keyboard shortcut often causes confusion. It looks like the execution of the command has ended.) The output does not resume until you press Ctrl+Q.

If your console responds to Ctrl+S in this way, you must first suppress the so-called flow control by using the stty -ixon command before you can use Ctrl+S as a search function.

7.3.3 Alias Abbreviations

alias

The alias command can save you some typing when you enter commands in the shell. This command is used to define abbreviations. When processing the command line, it checks if the first word contains an abbreviation. If so, the abbreviation is replaced with the full text.

Abbreviations for a certain combination of options or for file names are not possible because bash does not search the other parameters of a command for abbreviations. However, bash recognizes special cases where multiple programs are named on a command line (pipes, command substitution, sequential execution of commands with ;) and searches all occurring command names for abbreviations.

The following command defines the abbreviation cdb, which allows me to quickly change to the directory I often need, /home/kofler/linuxbook:

```
user$ alias cdb='cd ~kofler/linuxbook'
```

alias calls can also be nested. Note that alias abbreviations take precedence over commands with the same name. This fact can be used to avoid the unwanted invocation of a command:

```
user$ alias more=less
```

From now on, any attempt to call the `more` command will result in the start of the more powerful `less` program. If for some reason you still need `more`, you must specify the entire pathname (`/bin/more`) or prefix it with a backslash (`\more`). The backslash prevents the alias analysis in this case.

Saving abbreviations permanently

`alias` abbreviations can be deleted again using `unalias`. Otherwise, they will remain valid until you leave the shell (i.e., until you log out at the latest). If you need certain shortcuts repeatedly, you should include the `alias` statements in the `/etc/bashrc` file (applies to all users) or the `.bashrc` file in your home directory (applies only to that user).

Many distributions have various predefined `alias` abbreviations. For example, if `rm` keeps asking if you really want to delete the file, this is usually caused by a predefined alias, `rm=rm -i`.

A list of all currently valid abbreviations is provided by the `alias` command. The following lines indicate where Debian, Fedora, SUSE, and Ubuntu consider `alias` definitions:

```
/etc/bashrc
/etc/profile.d/*.sh
.bashrc
.alias  (SUSE only)
```

Custom shell scripts that are stored in a directory contained in `PATH` can have a similar effect to abbreviations. A script has the advantage over an alias in that it can process parameters.

7.4 Input and Output Redirection

When you run commands in `bash`, there are three standard files. The term *file* can cause a little confusion here. These are not actually real files, but file descriptors that are treated like files at the operating system level:

- **Standard input**
  The currently executing program, such as `bash` or any command started from there, reads all input from standard input. If you work interactively in a terminal window, then the keyboard is considered the default input source.
- **Standard output**
  The entire output of the program is directed here—for example, the listing of all files via `ls`. Usually, the terminal window is the standard output.

- **Standard errors**
  Error messages are also usually displayed in the current terminal.

Redirection

In itself, all this is self-evident: Where else but from the keyboard should the inputs come? Where else but on the screen should results or errors be displayed? However, the ability to redirect the standard input or output is noteworthy. For example, the case may arise that the table of contents of the current directory is not to be displayed on the screen, but saved in a file. This means the standard output is supposed to be redirected to a real file. This can be done in `bash` by using the `>` character:

```
user$ ls *.tex > content
```

The `content` text file now contains a list of all `*.tex` files in the current directory. This type of output redirection is certainly the most common method. However, there are many other variants (see also Table 7.3):

- `2> file` redirects all error messages to the specified file.
- `>& file` or `&> file` directs both the standard output and all error messages to the specified file.
- If instead of `>` the duplicate `>>` is used, then the respective outputs will be appended to the end of an already existing file.

Command	Function
`command > file`	Directs standard outputs to the specified file
`command < file`	Reads input from the specified file
`command 2> file`	Redirects error messages to the specified file
`command >& file`	Redirects outputs *and* errors
`command &> file`	Also redirects outputs *and* errors
`command >> file`	Appends standard output to the existing file
`command &>> file`	Appends output and errors to the file (as of `bash` 4.0)
`command1 | command2`	Forwards output from command 1 to command 2
`command | tee file`	Displays the outputs and saves a copy at the same time

Table 7.3 Input and Output Redirection

You can perform an input redirection by using `< file`. Commands that expect input from the keyboard then read it from the specified file.

Warning

It is not possible to edit a file and write the result back to this file at the same time! `sort file > file` or `sort < file > file` causes `file` to be deleted!

The `sponge` command from the `moreutils` tool collection, which can be installed as a package in many distributions, provides a solution here. `moreutils` contains various commands that are useful both for the interactive use of `bash` and for script programming. An amazing number of commands have to do with input and output redirection in various shapes, just like `sponge`.

Pipes

Pipes are formed using the | character. The output of the first command is used as input for the second command. In practice, you will often form pipes together with the `less` command when you want to view longer outputs page by page.

The following command determines the table of contents of the current directory and writes it to a pipe. From there, the `less` command executed in parallel reads its input and displays it on the screen:

```
user$ ls -l | less
```

Pipes are also excellent for combining different commands. Thus, the following command returns a sorted list of all installed RPM packages:

```
user$ rpm -qa | sort
```

FIFO

Instead of pipes, so-called FIFO files can also be used for input and output redirection. FIFO stands for *first in, first out* and implements the idea of a pipe as a file. The input of FIFO files is much more complicated than that of pipes, but they make it clear what the | character actually does. In practice, they are used so that two independent programs can communicate with each other.

The following three commands first set up a FIFO file, then `ls` is started as a background process, which writes its outputs to the file. From there, `less` reads the data again and displays it on the screen:

```
user$ mkfifo fifo
user$ ls -l > fifo &
user$ less < fifo
```

Only commands that read the commands to be processed from the default input channel are suitable for formulating a pipe. If this is not the case, you can achieve similar effects by a command substitution or by the `xargs`

command. These and other substitution mechanisms are described in Section 7.6.

7.4.1 Output Multiplication Using tee

tee

It sometimes happens that the output of a program should be saved in a file, but you still want to follow the progress of the program on the screen. In this case, a duplication of the output is required, whereby one copy is displayed on the screen and the second copy is saved in a file. This task is performed by the `tee` command:

```
user$ ls | tee filelist
```

The table of contents of the current directory is thus displayed on the screen and simultaneously stored in the `filelist` file. The standard output is first forwarded to the `tee` command. By default, this command displays the standard output in the terminal and saves the copy of it in the specified file. You will notice that this is really a multiplication of the output when you also forward the standard output of `tee` to a file:

```
user$ ls | tee filelist1 > filelist2
```

The result consists of two identical files, `filelist1` and `filelist2`. The preceding command is purely exemplary. Somewhat more difficult to understand, but more meaningful, is the following example:

```
user$ ls -l | tee filelist1 | sort -k 5 -n > filelist2
```

In `filelist1`, there is again the "normal" table of contents, which was automatically sorted by file names by `ls`. The copy of this output was passed to `sort`, sorted there by file size (fifth column), and stored in `filelist2`.

7.5 Executing Commands

Usually you start commands simply by entering the command name. In addition to that, there are some options to execute multiple commands in a row (see Table 7.4).

Command	Function
`cmd1; cmd2`	Executes the commands one after the other
`cmd1 && cmd2`	Executes command 2 if command 1 was successful

Table 7.4 Executing Commands

Command	Function
`cmd1 || cmd2`	Executes command 2 if command 1 returns an error
`cmd &`	Starts the command in the background
`cmd1 & cmd2`	Starts command 1 in the background, command 2 in the foreground
`(cmd1 ; cmd2)`	Executes both commands in the same shell

Table 7.4 Executing Commands (Cont.)

Scripts in the Home Directory

Scripts or programs stored in the currently active directory cannot be executed easily for security reasons. Rather, you must prefix the script name with `./`, such as `./myscript`, where `.` is an abbreviation for the current directory.

Background processes

The most important and most frequently used special character is `&`. When entered at the end of the command line, `bash` starts this program in the background. This is especially useful for time-consuming programs, because work can be continued immediately:

```
user$ find / -name '*sh' > result &
[1] 3345
```

The preceding command searches the entire file system for files ending with the letters *sh*. The list of files will be written to the `result` file. Because the command is executed in the background, you can continue working immediately. The output `[1] 3345` means that the background process has the process identification (PID) number 3345. The PID number is of interest if the process is to be terminated prematurely via `kill`. The number in square brackets indicates the number of the background process started in `bash` and is usually not of interest.

If you forget the `&` character when starting a command, you do not need to wait, nor do you need to forcibly stop the program via Ctrl+C. Rather, you should interrupt the program using Ctrl+Z and continue it as a background process via `bg`.

Executing multiple commands

After the `&` character, you can also enter another command. In this case, the first command will be executed in the background, while the second one

gets executed in the foreground. In the following example, the `find` command is started again in the background. At the same time, however, the current table of contents is output via `ls`:

```
user$ find / -name '*sh' > result & ls
```

If you specify a semicolon instead of the `&` character, `bash` executes the commands one after the other and in the foreground:

```
user$ ls; date
```

The preceding command first displays the current table of contents and then outputs the current date. If all this information is to be redirected to a file via `>`, you need to put both commands in parentheses. This means that both commands are executed by a single shell:

```
user$ (ls; date) > content
```

The `content` file now contains the file list created by `ls` and the current date determined via `date`. The parentheses cause the two commands to be executed within a new shell (a *subshell*) and therefore also provide a common result.

Conditional command execution

You can use the character combinations `&&` and `||` to execute commands conditionally—that is, in dependence on the result of another command:

```
user$ command1 && command2
```

The preceding command executes `command 1`. Only if this command was successful (no error; return value 0) will `command 2` be executed afterward. A typical example (that applies to Debian and Ubuntu) is first updating the package sources and then performing an update without prompting:

```
root# apt update && apt full-upgrade -y
```

In practice, the OR combination of two commands is less common:

```
user$ command1 || command2
```

This executes `command 1`. Only if an error occurs during the execution of this command (return value unequal 0) will `command 2` be executed afterward as well.

The `if` command provides further options for creating conditions and branches, although this is only of interest for shell programming.

7.6 Globbing and Substitution/Expansion

This section deals with various mechanisms used by `bash` to process an expression composed of special characters into a new result. Consider the following examples:

- **Globbing**
  If you type "cmd *.txt", then `bash` searches all files corresponding to this pattern, such as `readme.txt` and `copyright.txt`, for example. What actually is executed then is `cmd readme.txt copyright.txt`. This means that `bash` replaced the search pattern with actual matching file names. This process is referred to as *globbing*.
- **Substitution/expansion**
  Say that there is a variable named `$name` that contains the string "Peter". If you now execute `echo "Hello, $name!"`, then `bash` replaces the variable with its contents and prints "Hello, Peter!". In addition to this simple parameter substitution, various other mechanisms exist. Depending on which documentation you consult, this mechanism is referred to as *substitution* or *expansion*. In this book, I stick with *substitution*.

In the following sections, I present the most important globbing and substitution mechanisms. This involves the use of various special characters (see Table 7.5). I discuss the parameter substitution mentioned earlier in more detail in Section 7.10.

Command	Function
`?`	Exactly one character
`*`	Any number (also zero) of arbitrary characters (but no .* files!)
`**`	All files and directories, also from all subdirectories (from `bash` 4.0 with `shopt -s globstar`)
`[abc]`	One of the specified characters
`[a-f]`	A character from the specified range
`[!abc]`	None of the specified characters
`[^abc]`	As above
`~`	Abbreviation for the home directory
`.`	Current directory
`..`	Parent directory

Table 7.5 Special Characters for Globbing and Substitution in bash

Command	Function
ab{1,2,3}	Returns ab1 ab2 ab3
a{1..4}	Returns a1 a2 a3 a4
$varname	Returns the content of the variable
$((3*4))	Arithmetic calculations; returns 12
`command`	Replaces the command with its result
$(command)	As above; alternative style
command "character"	Prevents the analysis of all special characters except $
command 'character'	As above, but even more restrictive (no variable substitution)

Table 7.5 Special Characters for Globbing and Substitution in bash (Cont.)

Globbing: file names with * and ?

If you type "rm *.bak" and the rm command actually deletes all files ending with .bak, then bash is responsible for that. The shell searches the current directory for matching files and replaces *.bak with the corresponding file names.

The ? (exactly one arbitrary character) and * (any number, including zero, of arbitrary characters) wildcard characters are permitted. The [abe-h]* string represents file names that start with one of the a, b, e, f, g or h characters. If ^ or ! is specified as the first character inside the square brackets, then all characters except the specified characters are allowed. ~ can be used as an abbreviation for the home directory.

You can easily test the function of special characters using the following echo command. The first command returns all files and directories in the root directory. The second command restricts the output to files and directories starting with the letters a-f:

```
user$ echo /*
/bin /boot /dev /etc /home /lib /lib64 /lost+found /mnt /opt /proc
/root /run /sbin /srv /sys /tmp /usr /var

user$ echo /[a-f]*
/bin /boot /dev /etc
```

Because the formation of the file names is not done by the respective program but by bash, the results sometimes look different than you would probably expect. Thus, ls * can result in an almost endless list of files, even if there are only a few files in the current directory. After expansion with *,

a list of all files and directories is passed to the `ls` command. `ls`, in turn, displays not simply the names of directories, but the entire contents of these directories. If you just want to have a simple list of all files and directories, you must use the `-d` option, which prevents displaying the contents of the directories that are in the parameter line.

If you want feedback on how `bash` works internally, you can run `set -x`. `bash` then displays the way the command line is parsed (with any preset options and with the file names expanded) before executing any further command.

By default, `*` does not take files or directories into account that start with a dot (i.e., those that are "hidden"). If you want to capture those as well, you need to use `shopt` to set the `dotglob` option in `bash`:

```
user$ shopt -s dotglob
user$ echo *
...
user$ shopt -u dotglob   # deactivate dotglob again
```

If you want to enable `dotglob` permanently, you must add the `shopt` line to `.bashrc`.

File name formation with **

The `**` character combination recursively covers all (sub)directories. To maintain compatibility with older `bash` versions and to protect against erroneous use of this character combination (there are often tens of thousands of subdirectories and files in `/`), this new feature, already introduced in version 4, is not active by default. If you want to use it (e.g., in a script), you have to use `shopt -s` to set the `globstar` option in `bash`:

```
user$ shopt -s globstar
user$ echo **
...
```

`**` provides a very simple way to recursively find or process (copy, delete, etc.) all files of a certain type. The following command determines the number of PDF files in the current directory and all subdirectories:

```
user$ ls **/*.pdf
```

Brace expansion: string formation with {}

`bash` assembles all conceivable character string combinations from character strings that are specified in curly brackets. The official name for this substitution mechanism is *brace expansion*. Thus, `part{1,2a,2b}` becomes `part1 part2a part2b`. Brace expansions can reduce the typing effort when accessing multiple similar file names or directories. Compared to wildcard characters like `*` and `?`, they have the advantage that file names that do not yet exist can also be formed (e.g., for `mkdir`):

```
user$ echo {a,b}{1,2,3}
a1 a2 a3 b1 b2 b3

user$ echo {ab,cd}{123,456,789}-{I,II}
ab123-I ab123-II ab456-I ab456-II ab789-I ab789-II
cd123-I cd123-II cd456-I cd456-II cd789-I cd789-II
```

You can formulate enumerations elegantly in the notation `{a..b}`, where `a` and `b` may optionally be numbers or letters. The following examples explain how this works better than any description:

```
user$ echo {1..5}
1 2 3 4 5

user$ echo {z..t}
z y x w v u t
```

Calculate arithmetic expressions: (( ))

Usually, `bash` is not able to perform calculations. If you enter `2+3`, the shell does not know what to do with this expression. If you want to perform a calculation within the shell, you need to enclose the expression in double parentheses and prefix it with a `$` character:

```
user$ echo $((2+3))
5
```

Inside the parentheses, most operators known from the C programming language are allowed: `+ - * /` for the four basic arithmetic operations, `%` for modulo calculations, `== != < <= > >=` for comparisons, `<<` and `>>` for bit shifts, `!&&` and `||` for logical NOT, AND, and OR, and so on. All calculations are performed for 32-bit integers (number range between +/-2147483648). If individual values are to be taken from variables, they must be preceded by a `$` character.

An obsolete but still acceptable notation for arithmetic expressions is `$[expression]`. An alternative way to perform calculations is provided by the `expr` command. This is a unique Linux command that works independently of `bash`.

7.6.1 Command Substitution

Command substitution allows you to replace a command within the command line with its result. To do this, this command must be enclosed between two ` characters. An alternative notation is `$(command)`. This latter notation is preferable, first because it reduces the confusion of using three different marks (", ', and `), and second because it can be nested.

Thus, the command marked in this way is replaced by its result. This substitution allows for the nested call of multiple commands, where one command passes its result to the other command. The following two equivalent commands illustrate this very powerful mechanism:

```
user$ ls -l `find /usr/share -name '*README*'`
user$ ls -l $(find /usr/share -name '*README*')
```

This command first executes `find /usr/share -name '*README*'`. The result of this command is a list of all files in the `/usr/share` directory that contain the string "README". This list is then inserted into the command line instead of the `find` command. The command line then reads, for example:

```
user$ ls -l ...  \
    > usr/share/doc/secureboot-db/README.Debian \
    > /usr/share/doc/sed/README ...
```

This command leads to the following result:

```
...
-rw-r--r-- 1    995 Dez  4  /usr/share/doc/secureboot-db/
README.Debian
-rw-r--r-- 1    731 Jan 17  /usr/share/doc/sed/README
...
```

This result would not be possible by using a simple pipe with the `|` character. `ls` does not expect any input via standard input and therefore also ignores the information that `find` provides via the pipe. The following command therefore simply displays the contents of the current directory; the results of `find` are not displayed:

```
user$ find /usr/share -name '*README*' | ls -l  # does not work!
```

xargs However, there is another solution that does not require command substitution. By using the `xargs` command, data from standard input will be passed to the command you specify after `xargs`:

```
user$ find /usr/share -name '*README*' | xargs ls -l
```

A major advantage of `xargs` is that there is no size limit for the data to be processed. If necessary, `xargs` calls the command multiple times and passes the data coming from the standard input in several steps. Command substitution, on the other hand, is limited by the maximum size of a command line—usually several thousand characters. Sharing file names causes problems if the file names contain spaces. You can work around these problems by passing the `-print0` option to `find` and the `--null` option to `xargs`. The following command sets the *execute* bit for all directories:

```
user$ find -type d -print0 | xargs --null chmod a+x
```

Even more find options

I have used `find` in the preceding examples as a starting point to show variations on how to process the result. In fact, `find` can do this itself. `find ... -ls` lists the `find` results in detailed notation, `find ... -exec ...` applies another command to the `find` results, etc. More information about `find` follows in Chapter 9, Section 9.3.3.

What is true for `find`, however, is not true for the majority of the other commands you usually deal with. And then you have to rely on `xargs` or the substitution mechanisms of `bash`.

7.6.2 Single versus Double Quotation Marks

'abc'

Because virtually every character in `bash` except letters and numbers has some special meaning, it seems almost impossible to use these characters in strings or file names. This problem can be solved in two ways: either the special character is preceded by a backslash \, or the entire string is enclosed in apostrophes or quotation marks. Thus, by specifying apostrophes, you can, for example, delete a file with the file name `ab* $cd`:

```
user$ rm 'ab* $cd'
```

Note the difference between straight apostrophes (i.e., ') to indicate strings and the accented character falling to the right (i.e., `, for command substitution!).

"abc"

Quotation marks have a similar effect to apostrophes. However, they are less restrictive and allow the interpretation of some special characters like $ \ and `. For this reason, in character strings that are enclosed in quotation marks, variables with a preceding $ character are parsed:

```
user$ echo "This is the access path: $PATH".
This is the access path: /home/kofler/.local/bin:/home/kofler/
bin:...
```

If single apostrophes are used instead of quotation marks, the entire string is output unchanged by `echo`.

7.7 Variables

Assignment operator

The functionality of `bash` and that of many other Linux programs is controlled by the state of variables. `bash` variables can only store strings. The assignment of variables is done by the assignment operator `=`. The easiest

way to display the contents of a variable is to use `echo`, prefixing the variable name with a `$` character:

```
user$ var='abc'
user$ echo $var
abc
```

Unconventional syntax

Strange rules apply to the handling of variables in `bash`. For reading variables, the name must be preceded by a dollar character. However, this does not apply to assignments. The preceding listing is therefore correct!

Also note that you do not specify a space between the variable name and the `=` operator when making assignments. `var = abc` is syntactically incorrect and will not work!

Assembling strings

Simple strings without special characters and spaces may also be formulated without apostrophes. The statements `a=3` and `a="3"` are therefore equivalent.

If you want to read the content of a variable in a string, you normally use the already mentioned substitution, such as `b="a is $a"`. Only if the variable names cannot be separated from the rest of the text do you have to use the somewhat more complicated `${varname}` notation:

```
user$ a=3
user$ b="\${a}xxx\${a}"
user$ echo $b
3xxx3
```

In the next example, the `bin` directory in the home directory is added to the predefined `PATH` environment variable. This allows for all scripts in this directory to be executed without an explicit path specification:

```
user$ echo $PATH
/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin
user$ PATH="$PATH:/home/kofler/bin"
user$ echo $PATH
/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/home/
kofler/bin
```

Mathematical expressions

Calculations with variables can be performed in the already presented notation with parentheses. The character strings contained in variables are evaluated as numbers. In addition, the dollar character may be omitted:

```
user$ a=3
user$ b=$((a*4))
user$ echo $b
12
```

Command substitution

If the result of a command is to be stored in a variable, the command substitution with `$(command)`, which has already been described as well, must be performed. In the following example, the current directory is stored in `a`:

```
user$ a=$(pwd)
user$ echo $a
/home/kofler
```

Defining variables permanently

The contents of variables are stored within a shell session and are lost when the shell is exited. If certain variables are constantly needed again, the assignments should be made in the `/etc/profile` file or in `.profile` in the home directory. These two files (if present) are automatically executed when `bash` is started.

7.7.1 Environment Variables

Usually, variables are not passed on to newly started processes. So if you assign `a=3` and then run `ls` or start a script, neither the command nor the script can read the variable. `a` is only usable within the local shell session.

There is one exception to this rule: Variables can be explicitly declared as *environment variables*. To do this, you need to call `export varname` or `declare -x varname`. Environment variables can then be analyzed by all commands or scripts started thereafter.

There are numerous commands for variable management within the shell, whereby there are functional overlaps. The following examples attempt to alleviate some of the confusion by using similar commands:

```
let a=3           # normal variable
declare a=3       # equivalent
a=3               # usual short notation, also equivalent
export a          # makes a the environment variable
export a=3        # combines the two previous statements
declare -x a=3    # equivalent to export a=3
set               # lists all variables
printenv          # lists only environment variables
export            # also lists environment variables
unset a           # deletes the variable
```

7.7.2 Predefined Environment Variables

In `bash`, some environment variables are predefined that have a special meaning. These variables are capitalized throughout. Be careful not to accidentally overwrite or incorrectly change any of these variables! The following list describes the most important variables in alphabetical order:

- **BASH**
  Contains the path to `bash`.
- **HOME**
  Contains the path of the home directory—for example, `/home/kofler`.
- **LOGNAME**
  Contains the login name (user name).
- **HOSTNAME**
  Contains the host name (computer name).
- **OLDPWD**
  Contains the path of the last active directory.
- **PATH**
  Contains a list of directories separated by colons. When `bash` is supposed to execute a command, it searches all directories listed in `PATH` for the desired command.

  `PATH` is set distribution-specifically at various points during the startup process. The best place to make changes that apply to all users of the computer is `/etc/profile`. If the changes only affect your own account, you should simply manipulate `PATH` in `.bashrc`:

  ```
  # addition in /etc/profile or in .bashrc
  PATH=$PATH:/myown/bin
  ```
- **PROMPT_COMMAND**
  Can contain a command that is executed each time before `bash` displays the command prompt.
- **PS1**
  Contains a string whose content is displayed at the beginning of each input line (prompt). Within the string, the following character combinations are provided, among others: `\t` for the current time, `\d` for the date, `\w` for the current directory, `\W` for the last part of the current directory (e.g., `ssl` in `/etc/cups/ssl`), `\u` for the user name, `\h` for the host name (computer name), and `\$` for the prompt character (`$` for regular users, `#` for `root`).
- **PS2**
  Like `PS1`, but the string is only displayed for multiline entries—that is, if the first line has been terminated using `\`. A typical setting is `">"`.
- **PWD**
  Contains the path of the current directory.
- **RANDOM**
  Returns a new random number between 0 and 32767 at each readout.

In addition to the variables described here, numerous other environment variables are normally defined that control functions of the shell as well as various other programs. You can get a list of all defined variables via `printenv | sort`.

7.8 Bash Scripts

Scripts are text files that contain Linux and `bash` commands.

Scripts can contain branches, loops and functions. This means that `bash` is a simple programming language. A typical application for scripts is the automation of frequently needed command sequences for installing programs, administrating the system, performing backups, configuring and running individual programs, and so on.

The following pages only provide an introduction to programming using `bash`. A large amount of further information and examples can be found on the website *https://linuxconfig.org/bash-scripting-tutorial* or in *Scripting: Automation with Bash, PowerShell, and Python* (2024), also published by Rheinwerk Publishing.

Finding scripts

Many commands and functions in Linux are based on scripts. The following `find/grep` command searches the `/etc/` directory for shell scripts. All files that are marked as executable and that contain the `\#! ... sh` string will be recognized:

```
user$ find /etc -type f -executable -exec \
  grep -q '#!.*sh' {} \; -print
```

7.8.1 Example 1: grepall

Let's suppose you frequently use the `grep` and `find` commands to search the current directory and all subdirectories for files that contain a certain string and have changed in the last seven days. The correct command looks as follows:

```
user$ find . -type f -mtime -7 -exec grep -q 'suchtext' {}  \; -print
```

If, like me, you puzzle over which combination of options is required each time, it makes sense to define the new `grepall` command, which takes over this task. To do this, start your favorite editor to write the `grepall` text file. The file consists of only two lines, the first of which specifies the program name of the interpreter that will execute the script file:

```
#!/bin/bash
find . -type f -mtime 7 -exec grep -q $1 {}  \; -print
```

shebang/hashbang

The first line of the bash script is called *shebang* or *hashbang* and starts with the `#!` character combination. This is followed by the path to the interpreter that will execute the following code. For this and the following examples, this results in the line `#!/bin/bash`.

Instead of `/bin/bash`, you can also specify `/bin/sh`. On many distributions, the script is then also executed by `bash` because `/bin/sh` is simply a link to `bash`. On Debian and Ubuntu, however, `/bin/sh` references the `dash` shell interpreter. This program is optimized for higher speed, but is not fully compatible with `bash`.

Execution bit

The attempt to execute the `grepall` file just created ends with an error message, *permission denied*. The reason for this message is that new files generally have the access bits (x) disabled to execute the file. But you can quickly change that using `chmod`. `grepall abc` then returns the desired list of all files that contain the string "abc":

```
user$ ./grepall abc
bash: ./grepall: Permission denied
user$ chmod a+x grepall
user$ ./grepall abc
./bashprg.tex
```

To run the `grepall` command without specifying the directory where the code is stored each time, you must move the file to a directory contained in `$PATH`. In many distributions, the `bin` directory within the home directory is provided for this purpose:

```
user$ mv grepall ~/bin
```

If the command is to be made available to all users of the computer, the `/usr/local/bin` directory is a good choice:

```
user$ sudo mv grepall /usr/local/bin
```

Creating Small Text Files without an Editor

If you want to avoid calling the editor, you can also create the `grepall` file using `cat`. First, enter the `cat > grepall` command. The command expects data from the standard input (keyboard) and writes it to the `grepall` file. Then enter the command with all its options. Finally, exit `cat` using Ctrl+D (this corresponds to end of file [EOF]). You can view the resulting file via `cat grepall`.

7.8.2 Example 2: stripcomments

The second example is also a one-liner. You pass a text file to the `stripcomments` command. The three nested `grep` commands now eliminate all lines that start with the `#` or `;` characters or are completely empty. Comments are also removed if there are space or tab characters before the `#` or `;` characters. The command is excellent for removing all comment lines from configuration files and displaying only the settings that are actually valid:

```
#!/bin/bash
grep -Ev '^[[:space:]]*#|^[[:space:]]*;|^$' $1
```

Here's why: The `^[[:space:]]*#` pattern finds lines starting with `#`, where there may be any number of spaces and tab characters between the start of the line (`^`) and `#`. Similarly, the `^[[:space:]]*;` expression captures all lines starting with `;`. The third pattern applies to empty lines consisting only of the beginning and end of line (`$`).

The `-v` option inverts the usual function of `grep`: instead of extracting the lines found, `grep` returns all lines to which the pattern does *not* apply. The `-E` option enables the extended `grep` syntax, which allows for combining multiple search expressions with the `|` character.

7.8.3 Example 3: applysedfile

The previous two examples show how you can save yourself some typing and thinking, but they do not even indicate the far-reaching possibilities of script programming. The next example offers more in this respect.

Let's suppose you are faced with the task of performing a series of similar find-and-replace runs in a whole bunch of files. The `applysedfile` script program can help you with such tasks. The call of this script looks as follows:

```
user$ applysedfile *.tex
```

This creates a backup copy `*.bak` of all `*.tex` files. Then the `sed` command is used to execute a whole list of commands for each `*.tex` file. These commands must be located in the `./sedfile` file, which is automatically used by `applysedfile`. The code of `applysedfile` looks as follows:

```
#!/bin/bash
IFS=$'\n'
for fn in $*; do
  echo "process $fn"
  # first create backup file
  cp "$fn" "${fn%.*}.bak"
```

```
  # then apply sed
  sed -f ./sedfile < "${fn%.*}.bak" > "$fn"
done
```

The script starts with a strange assignment to the `IFS` variable. This *internal field separator* determines at which characters the bash separates words—by default, at tabs, spaces, and line breaks. However, if there is now a `file with blank.tex` file, the file name variable `fn` passes through `file`, `with`, and finally `blank.tex` in order. The subsequent commands `cp` and `sed` then trigger errors because these files do not exist.

`IFS=$'\n'` causes word separation to occur only on a newline character. This is true for the file list passed with `*.tex`. The preceding dollar character ensures that `\n` is parsed correctly and the script thus also processes files with spaces correctly. `for` initiates a loop. For each loop pass, a file name is inserted into the `fn` variable. The list of file names comes from `$*`. This character combination is a placeholder for all parameters and file names passed to the program.

The loop body outputs the name of each file. `cp` creates a backup copy of the file. First, all characters starting from the first dot in the file name are deleted. Then `.bak` is appended.

Finally, the `sed` command is executed for the file, using the `sedfile` control file from the local directory.

To correct typos that happen to me regularly, I have a file that starts as follows:

```
s,athome,at home,g
s,tohelp,to help,g
s,everytime,every time,g
s,without furtherado,without further ado,g
...
```

This is a `sed` command for each line, which replaces the first string with the second one (command `s`). The trailing letter `g` means that the command should also be executed multiple times within one line, if the error in question should occur several times within one line.

7.8.4 Example 4: Backup Script

The following script is automatically executed on my root server every week:

```
#!/bin/bash
m=$(date "+%m")
cd /var
tar czf - www | curl -T - --netrc-file pw.txt \
  ftp://1.2.3.4/www-$m.tgz
mysqldump -u cms -pxxxx cms | curl -T - --netrc-file pw.txt \
  ftp://1.2.3.4/cms-$m.sql
```

First, the `m` variable is initialized, which contains the current month as a number. The date command returns the current date and time. The format string `+%m` extracts the month from it.

Now `tar` creates a backup of the `/var/www` directory. The archive is not saved directly to a file, but redirected to the `curl` command using `|`. `curl` transfers the data to an FTP server (IP address 1.2.3.4), reading the login name and password from the `pw.txt` file. On the FTP server, the backup is stored under the name `www-MM.tgz`, where `MM` indicates the month (`01` to `12`).

In this way, monthly backup versions are created over the course of a year, so that I can reconstruct an older state of my website if needed. At the same time, the space required for the backup files is small. At any given time, there is a maximum of 12 versions: `www-01.tgz` through `www-12.tgz`.

The `mysqldump` command creates a backup of the MySQL database `cms`, where the content management system (CMS) of my website stores all pages and countless other data. Again, the backup is passed to `curl` using `|` and stored on my FTP server.

I saved the entire script as `/etc/myscripts/backup`. The daily call is taken care of by Cron (see Chapter 10, Section 10.6). The matching configuration file `/etc/cron.d/backup` looks like this:

```
# every Sunday at 3:15 am
15 3 * * 0 root /etc/myscripts/backup
```

7.8.5 Example 5: Creating Thumbnails

Thumbnails are reduced versions of image files.

The following script is called via `makethumbs *.jpg`. It creates the `400x400` subdirectory where it stores reduced copies of the original images. The maximum size of the new images is 400 × 400 pixels, preserving the proportions of the original image. Images that are smaller remain unchanged, so they are not enlarged.

The `for` loop analyzes `$*`. This character combination allows the evaluation of all parameters passed to the program. The script applies the `magick`

command from the imagemagick package. The -resize option is responsible for the reduction. -size only causes faster processing:

```
#!/bin/bash
IFS=$'\n'
if [ ! -d 400x400 ]; then     # create subdirectory
  mkdir 400x400
fi
for filename in $*;  do    # process all JPEG files
  echo "processing $filename"
  magick -size 400x400 -resize 400x400 "$filename" \
  "400x400/$filename"
done
```

7.8.6 Example 6: Setting Up Student Accounts

I needed the following script when I wanted to set up an account for all students at the beginning of a Linux course. The starting point was the students.txt file, which contained the last names line by line:

```
huber
mueller
schmidt
...
```

Now, I could have set up each student individually, either with a graphical user interface or by manually running useradd, passwd, and chage. Instead, I wrote the following miniscript:

```
#!/bin/bash
while read username; do
  pw="$username-1234"
  echo "Account: $username  Password: $pw"
  useradd $username
  echo "$username:$pw" | chpasswd
  chage -d 0 -E 2024-12-31 $username
done < students.txt
```

The script runs through the names of all students in the while loop. The current name is located in the username variable. The pw variable contains the initial password, which consists of the name plus -1234, such as huber-1234. useradd sets up the account. echo prints the account name and password. Of course, this output is not displayed, but is redirected to the chpasswd command via |.

`chage -d 0` makes sure that every student has to change his or her password immediately after the first login. The `-E 2024-12-31` option has the effect that the accounts expire at the end of December 2024 and can then no longer be used. To delete the accounts together with the assigned `/home` directories later, there is another script:

```
#!/bin/bash
while read username; do
  userdel -r $username
done < students.txt
```

7.8.7 Example 7: Changing Multiple MySQL/MariaDB Databases

In my role as a database administrator, I occasionally have to make changes to a lot of databases of the same type, such as adding a column to a table in 50 customer databases of the same structure. If I wanted to do this manually, I would have to create a connection to each database using `mysql dbname` and then run `ALTER TABLE tblname ADD ...`.

But just three lines of `bash` code are much more efficient:

```
#!/bin/bash
for db in $(cat /etc/mydbs.txt); do
  mysql $db < updates.sql
done
```

The script runs through all databases listed line by line in the `/etc/mydbs.txt` file. For each database, `mysql` executes the SQL commands stored in the `updates.sql` file.

The only requirement is that the user running the script can connect to all databases without explicitly entering a password. For this to work, the MySQL-specific `.my.cnf` or `.mylogin.cnf` file must contain the required authentication data.

7.9 Basic Rules for Bash Scripts

Shebang/hashbang

As I mentioned when describing the examples from the previous section, `bash` scripts must start with the line `#!/bin/bash`. This line, the so-called shebang or hashbang, specifies by which interpreter the following code is executed—in the examples in this chapter, by `bash`.

Execute bit

Shell scripts can only be executed if the access bits for read access (`r`) and execution (`x`) are set. The `r` bit is set by default for new files anyway, but you have to take care of the `x` bit (for *execute*) yourself once:

```
user$ chmod +x myscript
```

bin directory

If you write a collection of your own shell script programs for daily use, it makes sense to store them in a central location.

The best directory to use is `~/bin`. If you then make the following change in `.profile`, these script programs can be run without a full path specification (in some distributions this is not necessary at all, because there `~/bin` is always part of `PATH`):

```
# addition in ~/.profile resp. in ~/.bashrc
PATH=$PATH':~/bin'
```

Ending a script prematurely (exit)

`exit` terminates the running script, and thus the script returns the return value of the last executed command. If you don't want this, you can explicitly pass a value to `exit`. The value 0 signals an error-free end to the program, 1 indicates a general error, and 2 indicates an error in the passed parameters. All other error codes are application-specific.

Behavior in case of an error

In most programming languages, the following holds true: as soon as an unhandled error occurs, the program terminates. Things are much more relaxed in `bash` scripts: the fact that errors occur during command execution—that is, that a command ends with an error code not equal to 0—is accepted as an everyday occurrence. The execution of the script simply continues with the next statement.

Obvious syntax errors represent an exception—for example, for a `for` loop without `done` or for a character string that begins with " but where the second " character is missing. In such cases, `bash` can no longer decode the remaining program structure: it displays an error message and aborts the code execution.

Even though the high error tolerance of `bash` is often convenient, sometimes you do want a script to abort at the first error and not produce possible subsequent errors. You can achieve this by including `set -e` in the code. From this position on, errors lead to the immediate end. If you want this strict error control to apply to the entire script, you need to specify the `-e` option right in the first line:

```
#!/bin/bash -e
# this script will abort at the first error
...
```

For linked commands, `set -e` or the bash option `-e` apply only to the overall result. If `cmd1` fails in the following example—even if it is because the

command does not exist—then `cmd2` will be executed. Only if this command also leads to an error will the script abort:

```
#!/bin/bash -e
cmd1 || cmd2
```

7.10 Variables in Bash Scripts

I have already provided some initial information on dealing with variables in Section 7.7. This section covers other aspects of variable management that are particularly relevant to shell programming. In detail, it is about the scope of variables, about some variables predefined in `bash` (e.g., `$*` or `$?`), about the mechanism of parameter substitution for analyzing and processing strings in variables, and finally about the input of variables in shell programs.

7.10.1 The Scope of Variables

Passing on variables

To run a command or a script, `bash` creates a new process with its own PID number. This is an internal Linux number to identify and manage the process. Of the variables, only those declared as environment variables (`export` or `declare -x`) are passed to the new process. When a command is started in the foreground, `bash` enters the background during its execution and waits for the command to finish.

Subshell

The processing of a script takes place not in the running shell, but in a subshell started especially for this purpose. So now there are two instances of `bash` running: one as its command interpreter and the second one to execute the script. This is useful in that the original shell and the script can run in the background without interfering with each other, if necessary.

The concept of subshells affects variable management in that each (sub) shell has its own variables. As is the case when any other program is started, only environment variables are passed to the subshell. After that, the variables in the two shells are independent of each other. Changing variables in one shell has no effect on variables in the other shell.

source command

Sometimes you want to declare new variables or permanently change existing variables using a shell program. To make this possible, you can also run a script within the current `bash`—that is, without automatically starting a subshell. To do this, you need to put a period and a space before the script's file name. This corresponds to the short notation of the shell command, `source`.

Let's look at an example. Say that you want to write a shell program that extends the PATH variable with the path of the current directory. The required addpwd program is quite simple:

```
#!/bin/bash
# shell program addpwd adds the current directory to the path
PATH="$PATH:$(pwd)"
```

The PATH variable thus stores the previous content of this variable, a colon, and finally the result of the pwd command via command substitution. The following test run proves that the content of the PATH variable in the current shell does not change until addpwd is started with a preceding dot. Within the subshell that was started when addpwd was first called, PATH is of course also changed—but this change only applies as long as addpwd is running:

```
user$ echo $PATH
/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin

user$ ./addpwd      # Caution: no change of PATH
user$ echo $PATH
/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin

user$ . addpwd      # this is how it works!
user$ echo $PATH
/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/home/user
```

7.10.2 Variables Predefined by the Shell

Within shell programs, some variables predefined by bash can be accessed. These variables cannot be changed by assignments, but only read. The name of the variable is formed by various special characters. In Table 7.6, the variables are specified with the preceding $ character.

Variable	Meaning
$?	Return value of the last command
$!	PID of the last started background process
$$	PID of the current shell
$0	Filename of the currently executed shell script (or of the symbolic link referencing the file)
$#	Number of parameters passed to the shell program

Table 7.6 $ Variables

Variable	Meaning
$1 to $9	Parameters 1 to 9
$* or $@	Totality of all passed parameters

Table 7.6 $ Variables (Cont.)

Parameter

On the use of these variables, note that $0 to $9, $#, and $* are used to analyze the parameters passed to the batch program. Almost every script example in this chapter shows possible applications for this.

In connection with the analysis of parameters, the bash command shift is quite interesting. This command virtually pushes the passed parameters through the ten variables $0 to $9. If you execute shift 9, the first nine parameters passed to the program are lost, but then the next nine parameters can be conveniently addressed. shift without further specification shifts the parameter list by one parameter.

Last result

$? can be used to create conditions to make the further course of the program dependent on the result of the last command.

Basically, it's also possible to specify a command directly as a condition in if. The $? variable has the advantage of avoiding overly long and confusing statements.

PID

The $$ variable contains the PID. This numerical value is used internally in Linux to manage the processes. The PID is unique; that is, there is never any second process with the same number in the entire system. That is why this value is excellent for creating a temporary file. For example, you can use ls > tmp.$$ to store a list of all files in the tmp.nnn file. Even if the same batch file is running in another terminal at the same time, there is certain to be no name conflict because of the different PIDs of the two shells.

7.10.3 Arrays

Besides simple variables, bash also uses arrays. Up to and including version 3, the index must be a number. Note the ${field[n]} syntax for accessing the *n*th element, which differs from C:

```
x=()                    # definition of an empty array
x[0]='a'                # assign array elements
x[1]='b'
x[2]='c'
x=('a' 'b' 'c')         # short notation for the previous four lines
echo ${x[1]}            # read an array element
echo ${x[@]}            # read all array elements
```

bash also supports associative arrays, where the array elements can be accessed by freely chosen strings instead of by passing index numbers. To do this, you must explicitly declare the field variable as associative by using declare -A! Otherwise, the variable is considered a normal field. The character strings used in the index are evaluated to 0, and you get an ordinary array consisting of only one element (index 0):

```
declare -A y                # definition of an empty associative array
y[abc]=123                  # assign an element of associative array
y[efg]="xxx"
y=( [abc]=123 [efg]="xxx" ) # Short notation for the above two lines
echo ${y[abc]}              # Read an array element
```

You can use mapfile to read an entire text file line by line into the elements of an array:

```
mapfile z < textfile
```

7.10.4 Parameter Substitution

Under the term *parameter substitution*, bash provides a number of commands that can be used to edit character strings stored in variables. Note that the variable name is specified *without* a preceding $ character. If, on the other hand, the comparison pattern is to be read from a variable, a $ character must be used there:

- **${var:-default}**
  If the variable is empty, the construct returns the default setting as a result; otherwise, it returns the content of the variable. The variable is not changed.
- **${var:=default}**
  The content of the variable is changed at the same time if it was previously empty.
- **${var:+new}**
  If the variable is empty, it remains empty. If, on the other hand, the variable is already being used, the previous content is replaced by a new setting. The construct provides the new content of the variables.
- **${var:?errormessage}**
  If the variable is empty, the variable name and the error message are output, and the shell program then gets terminated. Otherwise, the construct returns the content of the variable.
- **${#var}**
  Returns the number of characters stored in the variable as the result (0 if the variable is empty). The variable is not changed.

- **${var:n}**
  Returns the character string stored in `var` from the nth character, with the count starting at 0.
- **${var:offset:len}**
  Skips `offset` characters and then returns `len` characters.
- **${var#pattern}**
  Compares the beginning of the variable with the specified pattern. If the pattern is recognized, the construct returns the contents of the variable minus the shortest possible text that matches the search pattern. If, on the other hand, the pattern is not found, the entire contents of the variable are returned. In the search pattern, the wildcard characters known for creating file names can be used (`* ? [abc]`). The variable is not changed in any case:

  ```
  user$ dat=/home/mk/book/book.tar.gz
  user$ echo ${dat#*/}
  home/mk/book/book.tar.gz
  user$ echo ${dat#*.}
  tar.gz
  ```

- **${var##pattern}**
  As the previous item, but now the largest possible string that matches the pattern is eliminated:

  ```
  user$ dat=/home/mk/book/book.tar.gz
  user$ echo ${dat##*/}
  book.tar.gz
  user$ echo ${dat##*.}
  gz
  ```

- **${var%pattern}**
  Like `${var#pattern}`, but now the pattern matching is done at the end of the variable content. The shortest possible string from the end of the variable is eliminated. The variable itself remains unchanged:

  ```
  user$ dat=/home/mk/book/book.tar.gz
  user$ echo ${dat&#x25;/*}
  /home/mk/book
  user$ echo ${dat&#x25;.*}
  /home/mk/book/book.tar
  ```

- **${var%%pattern}**
  As the previous item, but the largest possible character string is eliminated:

```
user$ dat=/home/mk/book/book.tar.gz
user$ echo ${dat%%/*}
      - no output -
user$ echo ${dat%%.*}
/home/mk/book/book
```

- **${var/find/replace}**
  Replaces the first occurrence of the `find` pattern with `replace`:

```
user$ x='abcdeab12ab'
user$ echo echo ${x/ab/xy}
xycdeab12ab
```

- **${var//find/replace}**
  Replaces every occurrence of the `find` pattern with `replace`:

```
user$ x='abcdeab12ab'
user$ echo echo ${x//ab/xy}
xycdexy12xy
```

- **${!var}**
  Returns the content of the variable whose name is contained in `var` as a character string:

```
user$ abc="123"
user$ efg=abc
user$ echo ${!efg}
123
```

7.10.5 Read Variables Using "read"

You can use the `bash` command `read` to process user input. As a rule, you first use `echo` to output a short text informing the user what input you expect—for example, y/n, a numerical value, and so on. In this context, the `-n` option is useful so that the entry is made immediately after the `echo` text and not in the subsequent line:

```
echo -n "Your name please: "
read name
echo "Hello, $name!"
```

7.11 Branches, Loops, and Functions

This section describes which `bash` keywords you can use to structure your code. You can create branches using `if` or `case`; loops using `for`, `while`, or `until`; functions using `function`; and so on.

7.11.1 If Branches

if

As in almost any programming language, you can formulate branches using `if`. The syntax is as follows:

```
if condition1; then
    command1a
    command1b
[ elif condition2; then
    commands2 ]
[ else
    commands3 ]
fi
```

A concrete example looks as follows:

```
if [ $# -ne 2 ]; then
    echo "Two parameters must be passed to the command!"
    exit 2  # error code for wrong parameters
else
    echo "Parameter 1: $1, Parameter 2: $2"
fi
```

The criterion for branching is the return value of the last command before `then`. The condition is met if this command returns 0. If `then` is specified in the same line (and not in the next line), then the command must be terminated by using a semicolon.

Reversed Logic

Note that in `bash`, the truth values for true (0) and false (not equal to 0) are defined in reverse compared to most other programming languages! Commands that are terminated properly return 0. Any value other than 0 indicates an error. Some commands return different error values depending on the error type.

7.11.2 Formulating Conditions

In `bash`, you are not allowed to formulate conditions using ordinary operators as in other programming languages, such as `if $a < 3; then`, for example. Instead, you must use the `bash` command `test` in order to formulate conditions in loops and branches. The correct wording of the introductory example is `if test $a -lt 3; then`, where `-lt` stands for *less than*.

Numbers, strings, and files

`test` is used for multiple tasks: to compare two numbers, to compare character strings, and to test whether a file exists and has certain properties. The following examples show some syntax variants:

- `test "$x"`
  Checks whether `x` is in use. The result is false if the string has 0 characters, otherwise it is true.
- `test $x -gt 5`
  Checks whether the `x` variable contains a numerical value greater than 5. If `x` does not contain a number, an error message occurs. Instead of `-gt` (*greater than*), the following relational operators can also be used: `-eq` (*equal*), `-ne` (*not equal*), `-lt` (*less than*), `-le` (*less equal*), and `-ge` (*greater equal*).
- `test -f $x`
  Checks whether a file with the name specified in `x` exists.

  If `test` is to be executed interactively in the shell, after the `test command`, the `$?` variable (return value of the last command) can be read using `echo`:

```
user$ a=20
user$ test $a -eq 20; echo $?
0
user$ test $a -gt 20; echo $?
1
```

Short notations [ ] and [[ ]]

In the `if` example, I used the short notation `[ expression ]` instead of `test expression`. Note the syntactically prescribed spaces within the square brackets:

```
user$ if [ $a -eq 20 ]; then echo "a=20"; fi
a=20
```

`bash` also uses a variant with two square brackets with some additional test options (see Table 7.7). Note, however, that these tests are not available in simpler shells (e.g., in `/bin/sh`)!

Test Expression	Meaning
`[[ zk = pattern* ]]`	True if the string starts with `pattern`
`[[ zk == pattern* ]]`	As above
`[[ zk ~= regex ]]`	True if the regular pattern is true

Table 7.7 Bash-Specific Test Variants

Test Expression	Meaning
`[[ bed1 && bed2 ]]`	True if both conditions are true (*and*)
`[[ bed1 || bed2 ]]`	True if one of the conditions is true (*or*)

Table 7.7 Bash-Specific Test Variants (Cont.)

7.11.3 Case Branches

Case distinctions are introduced by the `case` keyword, followed by the parameter to be analyzed. After the `in` keyword, you need to specify sample strings with which the parameter will be compared. In this context, the same wildcard characters as for file names are allowed. The pattern is terminated with a parenthesis, `)`—that is, for example, with `--*)` to recognize strings starting with two minus signs. You can separate alternative patterns using `|`.

The commands that follow the parenthesis must be terminated by two semicolons. If an `else` branch is needed, then `*` must be specified as the last pattern; all strings match this pattern. When processing a `case` construct, only the first branch where the parameter matches the specified pattern is considered:

```
echo -n "yes/no? "
read answer
case $answer in
    y | yes ) echo "yes";;
    n | no  ) echo "no";;
    *)        echo "invalid input";;
esac
```

7.11.4 For Loops

`bash` uses three commands for creating loops: `for` executes a loop for all elements of a specified list. `while` executes a loop until the specified condition is no longer fulfilled, whereas `until` executes it until the condition is fulfilled for the first time. All three loops can be exited prematurely by using `break`. `continue` skips the remaining loop body and continues the loop with the next loop pass.

In the first example, the `i` variable is assigned the strings `a`, `b`, and `c` in sequence. In the loop body, the content of the variable is output between `do` and `done`. Note that a semicolon is required at the end of the list as well as at

the end of the `echo` command. These semicolons can only be omitted if the input is distributed across multiple lines (which often happens in script files):

```
user$ for i in a b c; do echo $i; done
a
b
c
```

The equivalent multiline formulation of the preceding command in a script file would look as follows:

```
#!/bin/bash
for i in a b c; do
  echo $i
done
```

Loop through files

The list for `for` can also be created using wildcards for file names or using `{..}` constructs to create strings. In the following example, all `*.tex` files are copied to `*.tex.bak` files. In the `cp` command, `$file` is always enclosed in quotation marks so that file names with spaces are also handled correctly:

```
user$ for file in *.tex; do cp "$file" "$file.bak"; done
```

Loop through all parameters

If you want to analyze the file names passed to a script (`$*` variable), you need to initialize the previously mentioned `IFS` variable with a newline character. Otherwise the analysis of `$*` will not work correctly for file names with spaces:

```
IFS=$'\n'
for file in $*; do
    cp "$file" "$file.bak"
done
```

Loop through all lines of a text file

You often need loops to process a text file line by line. That's no problem: simply pass the result of `cat file` to the `in` keyword! The following miniprogram creates a compressed backup in the `db.sql.gz` file for all databases named line by line in the `dbs.txt` file:

```
#!/bin/bash
# Loop through
for db in $(cat dbs.txt); do
  mysqldump $db | gzip -c > $db.sql.gz
done
```

Numeric for loops

The simplest way to formulate loops through a range of numbers is as follows:

```
for i in {1..12}; do
  echo "$i"
done
# Output 1, 2, 3, ..., 12
```

If you need leading zeros, you must specify them in the first parameter in `{n1..n2}`:

```
for i in {01..12}; do
  echo "$i"
done
# Output 01, 02, 03, ..., 12
```

For loops with a given increment, you need to use the notation `{n1..n2..step}`—so, for example:

```
for i in {1..12..3}; do
  echo "$i"
done
# Output 1, 4, 7, 10
```

Finally, you can also resort to the classic C syntax:

```
for ((i=1;i<=12;i++)); do
  echo "$i"
done
# Output 1, 2, 3, ..., 12
```

Note that `bash` does not provide any floating-point arithmetic. So all constructs presented here work only for integers.

7.11.5 While Loops

while

In the following example, the `i` variable is assigned the value 1. Then the variable in the loop body between `do` and `done` is incremented by 1 until the value 5 is exceeded. Note that conditions must be specified using the `test` command or its short form in square brackets, as with `if` branches:

```
user$ i=1; while [ $i -le 5 ]; do echo $i; i=$(($i+1)); done
1
2
3
4
5
```

The following loop processes all file names resulting from the `ls *.jpg` command:

```
ls *.jpg | while read file
do
  echo "$file"
done
```

7.11.6 Until Loops

until

The only difference between `until` loops and `while` loops is that the condition is formulated in a logically negated way. The following command is therefore equivalent to the `while` loop presented previously. Here, `-gt` is used to formulate the condition `i>5` (*greater than*):

```
user$ i=1; until [ $i -gt 5 ]; do echo $i; i=$(($i+1)); done
```

7.11.7 Functions

function

The `function` keyword defines a function that can be called in the script like a command. The code of the function must be enclosed in curly brackets. Within the function, local variables can be defined by using `local`. Functions can be called recursively. Functions must be declared *before* their first call!

Parameters can be passed to functions. Unlike many programming languages, the parameters are not enclosed in parentheses. Within the function, the parameters can be taken from the `$1` and `$2` variables, which means that a function processes parameters in the same way as a `bash` script. The following miniscript outputs *Hello World, Linux!*:

```
#!/bin/bash
function myfunc {
  echo "Hello World, $1!"
}
myfunc "Linux"
```

The `function` keyword is optional. If you omit `function`, however, the function name must be followed by parentheses. Thus, the following program is equivalent to the preceding example:

```
#!/bin/bash
myfunc() {
  echo "Hello World, $1!"
}
myfunc "Linux"
```

7.11.8 Heredoc Syntax

The Heredoc syntax allows you to formulate multiline constructs ending with a freely chosen string. This syntax is used particularly often to create multiline files or to define multiline variables:

```
# writes a multiline text to the myfile file
cat <<EOF > myfile
Line 1
Line 2
EOF

# stores a multiline text in the $myvar variable
read -r -d '' myvar <<EOF
Line 1
Line 2
EOF
echo "$myvar"
```

Instead of `EOF`, you can use any other string. It is customary to specify this string in uppercase characters. In the `read` command, the `-r` option causes the character string to be accepted unchanged (*raw*). `-d ''` (*delimiter*) means that the end of the line does not count as the end of the input. For the output of multiline character strings via `echo`, it's important to enclose the variable in quotation marks; otherwise, the line breaks will be lost.

7.12 Important Special Characters in Bash: Quick Reference

Table 7.8 summarizes all the special characters I introduced in this chapter.

Character	Meaning
;	Separates multiple commands
:	Shell command that does nothing
.	Insert shell code here (`. file` corresponds to `source file`)
#	Introduces a comment
#!/bin/bash	Identifies the desired shell for the shell program
&	Executes the command in the background (`com &`)
&&	Conditional execution of the command (`com1 && com2`)

Table 7.8 Special Characters in Bash

Character	Meaning
`&>`	Redirection of standard output and errors (corresponds to `>&`)
`|`	Creates pipes (`com1 | com2`)
`||`	Conditional execution of the command (`com1 || com2`)
`*`	Wildcard character for file names (any number of characters)
`?`	Wildcard character for file names (any character)
`[abc]`	Wildcard character for file names (one character from `abc`)
`[ expression ]`	Short notation for `test expression`
`[[ expression ]]`	As above, but additional, `bash`-specific syntax variants
`(...)`	Execute commands in the same shell (`(com1; com2)`)
`{...}`	Group commands
`{ , , }`	Assemble strings (`a{1,2,3}` → `a1 a2 a3`)
`{a..b}`	Assemble strings (`b{4..6}` → `b4 b5 b6`)
`~`	Abbreviation for the home directory
`>`	Output redirection to a file (`com > file`)
`>>`	Output redirection; append to existing file
`>&`	Redirection of standard output and errors (corresponds to `&>`)
`&>>`	Append standard output and errors to file (as of `bash` 4.0)
`2>`	Redirection of standard error output
`<`	Input redirection from a file (`com < file`)
`<< end`	Input redirection from active file to `end`
`$`	Marking of variables (`echo $var`)
`$!`	PID of the last started background process
`$$`	PID of the current shell
`$0`	File name of the currently executed shell script
`$1` to `$9`	The first nine parameters passed to the command

Table 7.8 Special Characters in Bash (Cont.)

Character	Meaning
`$#`	Number of parameters passed to the shell program
`$*` or `$@`	Totality of all passed parameters
`$?`	Return value of the last command (0 = OK or error number)
`$(...)`	Command substitution (`echo $(ls)`)
`${...}`	Various special functions for editing strings
`$((...))`	Arithmetic analysis (`echo $((2+3))`)
`"..."`	Prevent analysis of most special characters
`'...'`	Prevent analysis of all special characters
`` `...` ``	Command substitution (`` echo `ls` ``)

Table 7.8 Special Characters in Bash (Cont.)

Chapter 8
Zsh (Shell)

For decades, bash was *the* default shell of almost all Linux distributions. In recent years, however, there has been some movement in the selection of shells. zsh (pronounced as *z-shell*) is not only gaining popularity among professionals but is now enabled by default by some distributions (such as Manjaro and Kali Linux). Fans of zsh cite the following benefits, among others:

- Better ("smarter") completion of commands and parameters
- Command history shared across multiple shells
- Extendable through modules
- More visual design options for the input prompt

In this short chapter, I want to show how you can switch from bash to zsh, explain the benefits of zsh, and introduce you to the popular Oh My Zsh extension. Throughout the chapter, I assume that you know what a shell is and that you know the basic functions of bash. If necessary, you should take another look at Chapter 8.

zsh also on macOS

Interestingly, zsh is also gaining popularity outside of Linux: current versions of macOS also rely on zsh, but Apple's motivation has less to do with functionality and more with licensing issues. zsh uses a BSD-style license, while more recent bash versions are based on GPL 3. Apple, on the other hand, avoids integrating open-source code with a GPL-3 license into its products.

Personal Assessment

I only switched from bash to zsh—admittedly a bit by accident—in 2020. I primarily wanted to take advantage of the cool Git features that the Oh My Zsh extension brings. The changeover was trouble-free, with few changes in the operation of the shell. However, zsh behaves better, is more refined, and is more mature than bash in many details. There is not one decisive argument, but it is the total of many little things that make zsh so attractive. In a nutshell: I don't want to go back to bash!

Risk of overconfiguration

Countless options and expansion modules lead to the temptation of playing around with configuration for days in search of the perfect setup, but has one big disadvantage: Usually you work on many systems, at least as an

administrator. The more elaborate your setup is, the more dependent you become on it and want to set it up on *each* of your systems. The time you might gain with your super setup is lost on the other hand by migrating the setup from one computer to the next.

Programming

Like `bash`, `zsh` can be used to program scripts. However, this is rather uncommon, apart from `zsh`-specific extensions. As a scripting language, `bash` will probably remain the standard, even though the syntax of `zsh` is the same as `bash` except for a few subtle differences. In any case, this chapter focuses on the interactive use of `zsh` and does not address programming.

8.1 Installation and Configuration

Finding your active shell

If you are unsure about which shell is used in your Linux distribution, you can simply output the contents of the `SHELL` environment variable. It contains the path to the active shell, such as `/bin/bash` or `/usr/bin/zsh`:

```
user$ echo $SHELL
/usr/bin/zsh
```

Installing and activating zsh

All common distributions provide a package for `zsh`. To switch to `zsh`, you first need to install this package, then use `which` to determine the path, and then run `chsh`. Note that `chsh` always applies only to the user executing the command. If there are multiple users on a computer, each of them can set and use their favorite shell:

```
user$ sudo apt/dnf/zypper install zsh
user$ which zsh
/usr/bin/zsh
user$ chsh -s $(which zsh)
Password: **********
```

`chsh` enters the shell you want in the last column of the `/etc/passwd` file. The change will not take effect until the next login, so you have to log out and log in again.

The `chsh` command may not be available. In this case, you must install the appropriate package beforehand, such as `util-linux-user` on Fedora or `passwd` on Debian/Ubuntu.

Configuration files

After switching to `zsh`, the command prompt will probably look different from what you are used to from `bash`. In addition, various functions of `bash`, such as self-defined aliases, may no longer work. The reason for this can be found in the configuration. While `bash` parses `.bashrc` and some other configuration files by default, the corresponding files of `zsh` have different

names (see Table 8.1) and are probably still partly empty in your case—unless you use a distribution like Manjaro with a great default configuration (see Figure 8.1).

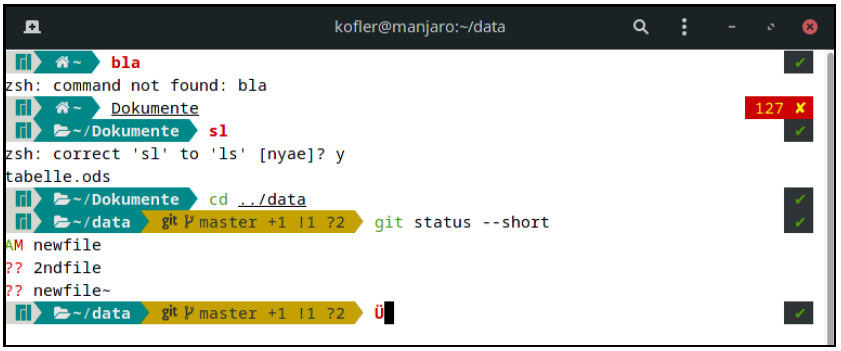


Figure 8.1 Zsh in Default Configuration on Manjaro: Displaying Current Directory and Status of Git Repository

Global	Local	Login Shell	Interactive Shell	Script
/etc/zshenv	.zshenv	•	•	•
/etc/zshrc	.zshrc	•	•	
/etc/zprofile	.zprofile	•		
/etc/zlogin	.zlogin	•		
/etc/zlogout	.zlogin	•		

Table 8.1 Different Zsh Configuration Files Considered Depending on Type of Shell

`zsh` parses the global configuration files first (first column), then the local files. Note that depending on the distribution, the global configuration files are not located directly in the `/etc` directory, but in the `/etc/zsh` directory. This is true for Debian and Ubuntu, for example.

A login shell exists if the shell is started directly during the login process, such as in a text console or via SSH. The shell is considered interactive if it accepts and responds to commands via a console, while the start did not occur as part of the login. Provided you work in a graphical user interface, this is the normal case.

First, you log in, then you start a terminal and run `zsh` in it. Table 8.1 shows that the two most important configuration files are `.zshenv` and `.zshrc`. `.zshenv` is primarily intended for defining environment variables that should also be available in scripts. In real life, the file often remains empty.

On the other hand, *.zshrc* is the right place to set options for interactive use of zsh, to define custom aliases, and so on.

Basic config using zsh-newuser-install

If zsh detects that there are no configuration files in your home directory yet, it starts the zsh-newuser-install configuration help:

```
This is the Z Shell configuration function for new users, zsh-
newuser-install.

You are seeing this message because you have no zsh startup files
(the files .zshenv, .zprofile, .zshrc, .zlogin in the directory ~).
This function can help you with a few settings that should make your
use of the shell easier.

You can

(q)  Quit and do nothing.  The function will be run again next time.
(0)  Exit, creating the file ~/.zshrc containing just a comment.
     That will prevent this function being run again.
(1)  Continue to the main menu.
(2)  Populate your ~/.zshrc with the configuration recommended
     by the system administrator and exit (you will need to edit
     the file by hand, if so desired).
```

If you choose option 2, the setup program copies the */etc/zsh/newuser.zshrc.recommended* file to .zshrc. The content of this default configuration varies depending on the distribution. Some distributions do not provide the file at all, in which case this option is missing.

If you want to use all the design options yourself, you have to choose variant 1 anyway. This will take you to another menu where you can configure various zsh functions step by step. I will focus on the most important items here:

- **History**
  This item is about whether zsh should remember the most recently executed commands, what the maximum number of these commands is, and where they should be stored. The default is to store a maximum of 1,000 commands in the .histfile file.
- **Completion**
  zsh acts extremely intelligently when you complete commands, options, files, and so on using the Tab key. Behind the scenes, the zsh component compsys is responsible for this. In the setup program, you can simply activate compsys with the default settings or perform a more detailed

configuration. If necessary, you can also start this later using the following two commands:

```
user$ autoload -Uz compinstall
user$ compinstall
```

- **Keys**
  To change the input in the command line, the Emacs keyboard shortcuts usually apply: Ctrl+A jumps to the beginning of the line, Ctrl+E to the end, and so on. Alternatively, you can also activate the usual keyboard shortcuts in the Vi editor here.
- **Options**
  In this menu item, you can switch some additional options on or off:
  - `autocd`
    If you enter the name of a directory and press Enter, `zsh` goes directly to that directory. It is not necessary to precede the `cd` command. `autocd` works only for directories whose name does not match that of a command.
  - `extendedglob`
    This option provides the `#` (imprecise search), `^` (negation), and `~` (exception rule) characters with a special meaning for globbing (i.e., for describing file names by patterns). Application examples follow in the next section. The option has the disadvantage that dealing with file names in which these three characters actually occur becomes more difficult. You must then enclose the corresponding expression in quotation marks.
  - `nomatch`
    This option causes `zsh` to throw an error if you specify a search pattern that does not match any file.

  More information about these and a lot more options is provided by `man zshoptions`.

If you decide to save at the end of `zsh-newuser-install`, your settings will be added to the `.zshrc` file. This file then looks similar to the following listing:

```
# .zshrc file
# History
HISTFILE=~/.histfile
HISTSIZE=1000
SAVEHIST=1000
# Completions
zstyle :compinstall filename '/home/kofler/.zshrc'
autoload -Uz compinit
compinit
```

```
# Emacs Keybindings
bindkey -e
# Options
setopt autocd extendedglob nomatch
```

Repeatedly calling zsh-newuser-install

Internally, `zsh-newuser-install` is not a script, but a function. If you want to run it again, you need to execute the following two commands:

```
user$ autoload -U zsh-newuser-install
user$ zsh-newuser-install -f
```

`zsh-newuser-install` cannot be executed by `root`.

Even more options

`zsh-newuser-install` gets you off to a good start, but it leaves out some useful features of `zsh`. The following listing contains some ideas that you can incorporate into your own `.zshrc` file if you want:

```
# Configuration ideas for .zshrc
setopt correct           # tries to correct typing errors

setopt sharehistory      # share and always update history file
                         # between multiple zsh processes immediately

setopt histignorealldups # do not store duplicates in history file
```

The `setopt <name>` command enables you to activate an option on a test basis, while `unsetopt <name>` allows you to deactivate it again. `setopt` without further parameters lists all currently active options. `man zshoptions` provides a reference of well over a hundred `zsh` options. The options are highlighted there with capital letters and underscores. Because `setopt` is not case-sensitive and ignores underscores, `setopt AUTO_CD` and `setopt autocd` are equivalent. If you want to be inspired further, you should take a look at the `zsh` configuration of various distributions as well as the sample files provided:

- **Debian/Ubuntu**
  File */etc/zsh/newuser.zshrc.recommended* or file */usr/share/doc/zsh/examples/zshrc*
- **Kali Linux**
  File *.zshrc* or */etc/zsh/newuser.zshrc.recommended*
- **Manjaro**
  The */usr/share/zsh* directory

Before you waste too much time with a manual configuration, you should try the Oh My Zsh extension. Many configuration requests, especially those relating to the design of the input prompt, then virtually solve themselves.

8.2 Usage

Basically, zsh works like any shell: you can execute commands, redirect the inputs and outputs using < and >, link commands using the pipe operator |, and so on. I have already presented these basic functions to you in detail in Chapter 7. For this reason, I describe only some features at this point, ones that are missing in this form in bash.

Autocorrect

If the correct option is set, zsh compares the command you enter with existing commands. If your incorrect entry differs only by a minor detail (e.g., a transposed letter as in the following example), the shell automatically suggests a suitable correction. The four options of nyae stand for *no* (execute command despite error), *yes* (correct the command), *abort* (abort the command), and *edit* (edit the command again). With *yes*, the command will be not only executed correctly, but also included correctly into the history:

```
user$ mdkir mydocuments
zsh: correct 'mdkir' to 'mkdir' [nyae]? y
```

Completion of name parts

As in bash, you can press Tab at any time in zsh to complete the current input. zsh is even more sophisticated than bash, especially when it comes to completing file names.

In bash, the completion function requires you to know the initial letters of the file or command. If zsh doesn't find a matching command or file, it will search for the specified letters further down the line. So if there is a file named linux-content.ods in your directory and you type ls content Tab, zsh will make it ls linux-content.ods. But this only works if there is no other file that starts with content.

Completing directories

zsh is also very clever when it comes to changing directories: In most cases, it's sufficient to enter the initial letters. To change to the directory with the text files for this book on my computer, I type cd n/l/t Tab Enter. Tab immediately turns the sparse directory specification into no-sync/linux/text. This assumes that there is no other directory that matches this combination of initial letters.

If the autocd option is active, the directory change works without the explicit cd command. Tab, however, must not be left out, as otherwise zsh will complain that the n/l/t directory does not exist.

Globbing

The term *globbing* refers to applying search patterns to file names. For example, ls *.pdf lists all PDF documents in the current directory. While bash does not perform globbing until the command is executed, you can initiate the process in zsh prior to that. To do this, you want to press Tab

after `ls *.pdf`. In the command line, `*.pdf` will then be replaced by the files that have been found. With `ls` this is rarely useful, but before `rm *.pdf` it is certainly a good idea to have a quick look at the names of the files to be deleted.

Qualifier

The trailing expression `(qualifier)` allows `zsh` to consider only the files that have a certain property for globbing (see Table 8.2).

Qualifier	For
`(/)`	Directories (no files)
`(.)`	Files (not directories)
`(@)`	Symbolic links
`(*)`	Executable files (oktal 100/010/001)
`(r)`	Files readable by the owner (400)
`(w)`	Files modifiable by the owner (200)
`(x)`	Files executable by owner (100)
`(s)`	setuid files (4000)
`(S)`	setgid files (2000)
`(md-2)`	Files that have been modified in the last two days
`(md+2)`	Files that have been modified more than two days ago
`(mh-6)`	Files that have been modified in the last six hours
`(md+6)`	Files that have been modified more than six hours ago
`(Lm+1000)`	Files larger than 1,000 MiB
`(Lk-250)`	Files smaller than 250 KiB

Table 8.2 Examples of Recursive Search for Specific Files

The following examples illustrate the qualifier syntax:

```
user$ ls *              (lists all files and directories)
user$ ls *(.)           (lists only files, no directories)
user$ ls *(/)           (lists only directories)
user$ ls *.tex(md-7)    (lists files that have been modified in the last week)
```

The `zsh` manual contains a list of countless other globbing qualifiers; see *http://zsh.sourceforge.net/Doc/Release/Expansion.html#Glob-Qualifiers*.

Recursive search

If you prefix a search pattern with **, zsh recursively searches all subdirectories according to the pattern. Thus, `ls **/*.pdf` takes into account files in the current directory as well as in all subdirectories. Of course, with large directory trees, this can take some time.

You can also specify a qualifier for the recursive search. `**/*(qualifier)` only considers files that have a certain attribute (see Table 8.2). The simple syntax avoids cumbersome `find` commands. This is especially useful for `chmod` commands where you often want to treat files and directories differently:

```
root# chmod 750 **/*(/)
root# chmod 640 **/*(.)
```

Alternatives

You can use `(variant1|variant2|variant3)` to formulate alternative globbing expressions. `ls *.(c|h)` thus lists all files ending with `.c` or `.h`.

Negation

If `extendedglob` is active, `^` negates the search expression. `ls -d ^*.pdf` thus lists all files or directories that do *not* end with `.pdf`. `^*.*` applies to all file names that do not consist of two parts (such as `Makefile`, but not `main.c`).

Exceptions

If `extendedglob` is active, you can use `~` to exclude individual files from the globbing pattern. `ls *.md~key.md` lists all markdown files except `key.md`. `ls *.md~sql?.md` refers to all markdown files except `sql1.md`, `sql2.md`, and so on.

Fuzzy search

Once again with `extendedglob`, the character combination `(#a<n>)` has a special meaning. It enables you to specify by how many errors the file name may deviate from the search expression. Missing characters, transposed characters, deviating characters and additional characters are considered as errors. Note that `(#a<n>)` is not a qualifier and must precede (not follow) the file name or search expression:

```
user$ ls concept.md          (exact match)
concept.md
user$ ls (#a2)concept.md     (fuzzy search with two errors)
concept.md  concept.pdf
user$ ls (#a3)concept.md     (fuzzy search with three errors)
concept.html  concept.md  concept.pdf  concept-v1.md
```

Expanding variables

Tab replaces not only globbing patterns with their corresponding file names, but also variable names with their contents. Thus, if you type "echo $PWD" and then press Tab, it becomes `echo /home/kofler/linux-book` or whatever your current directory is.

8.3 Oh My Zsh

There are countless extensions available for zsh. These can be divided into two groups: plugins that provide additional functionality and themes that change the look of command prompts.

The zsh script with the funny name *Oh My Zsh* helps you to add such extensions to the zsh configuration in an uncomplicated way. A wide range of plugins and themes is already included in the standard delivery, while others can be installed and integrated later. Besides Oh My Zsh, there are other plugin management systems for zsh, but I won't cover them here.

Installation

To install it, you need to download and run a small script from the project page at *https://github.com/ohmyzsh/ohmyzsh*. The script assumes that the git command from the package of the same name has already been installed. Instead of typing the following command, it's better to copy the code from the GitHub page:

```
user$ sh -c "$(curl -fsSL https://raw.githubusercontent.com/\
        ohmyzsh/ohmyzsh/master/tools/install.sh)"
```

During the installation, .zshrc will be overwritten. The previous content ends up in the backup file named .zshrc.pre-oh-my-zsh. From now on, Oh My Zsh will check for updates at startup. Accordingly, when you open a new terminal window, you are often prompted to update Oh My Zsh.

Basic configuration for Git users

By default, Oh My Zsh is characterized by a relatively moderate basic configuration. The prompt is predefined by the robbyrussell theme. As the only active plugin, git contributes countless aliases that help with the operation of git. alias lists all the abbreviations. The plugin also influences the prompt: when you use cd to change to a directory under the control of this version control system, it displays the information about the Git status as well as the name of the currently active branch.

Changing the configuration

To regard Oh My Zsh purely as a tool for Git enthusiasts falls far short of the mark. The real power of Oh My Zsh is that by changing a few lines in .zshrc, you can enable more plugins as well as a theme of your choice. For this purpose, all you have to do is change the settings of the ZSH_THEME and plugins variables. The following listing shows the syntax that you must adhere to. In particular, you must not separate the plugins with commas. Spaces are sufficient:

```
# .zshrc file
...
ZSH_THEME="robbyrussell"
...
plugins=(git dirhistory)
```

For the changed configuration to take effect, you must start a new shell, and the easiest way to do this is to open a new terminal window. By the way, the `dirhistory` plugin mentioned in the previous listing provides the Alt+← and Alt+→ keyboard shortcuts, which allow you to quickly jump through the last 20 directories used.

Themes and prompts

Themes control the color scheme of `zsh` and especially the design of the prompt. `zsh` provides an almost infinite number of design options in this regard. Instead of dealing with this yourself, it's much easier to try out one of the many predefined themes. On the following page, you will find screenshots of the many included themes: *https://github.com/ohmyzsh/ohmyzsh/wiki/Themes*.

There are also other themes that are not directly integrated into Oh My Zsh, such as Powerlevel10k (see Figure 8.2). You must first install these themes manually before you can activate them in `.zshrc`. Note that with some themes you also need custom fonts to display various special and status characters in the prompt. Copy the font files to the `.fonts` directory, then open the configuration dialog of your terminal program and set the appropriate font.

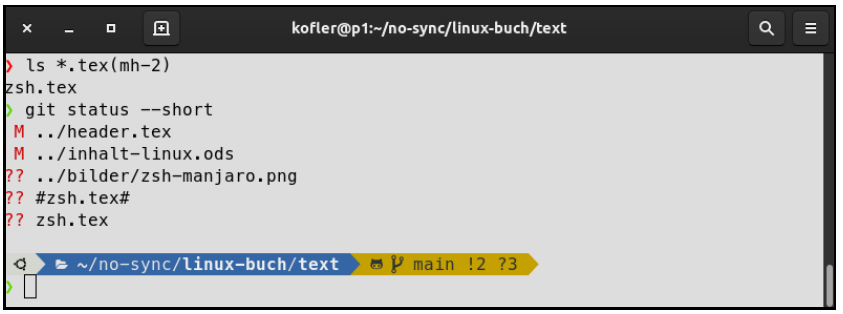


Figure 8.2 Zsh Terminal with Oh My Zsh Plugin and Powerlevel10k Theme

A detailed description of Powerlevel10k can be found on the project's GitHub page at *https://github.com/romkatv/powerlevel10k*. Do I still need to mention that Powerlevel10k itself again offers countless configuration options? I have already warned you that in `zsh` the real work is left behind because of all the tuning.

Plugins

An almost endless number of plugins for all conceivable tasks, tools and programming languages can be found at *https://github.com/ohmyzsh/ohmyzsh/tree/master/plugins*. A better guide is the following top 20 list, though it was compiled in 2018: *https://safjan.com/top-popular-zsh-plugins-on-github*.

Syntax highlighting

Each editor can highlight different parts of a program code. This must also work in the shell, right? I will briefly introduce three corresponding extensions here.

The `colored-man-pages` plugin provided by Oh My Zsh formats `man` pages using more colors.

The `colorize` plugin, also included in Oh My Zsh, provides the `ccat` and `cless` commands. When you use it to view text files, the commands try to recognize the syntax and highlight text in a meaningful way. Before adding `colorize` to the `plugin` line of `.zshrc`, you must install the `pygments` package or `python-pygments`, depending on your distribution.

The `ccat` and `cless` commands are aliases that refer to the `zsh` functions `colorize_cat` and `colorize_less`. They are noticeably slower than `cat` or `less`, which is why the obvious `alias less=cless` statement is not recommended.

A completely different approach is taken by the `zsh-syntax-highlighting` plugin, which is not included in Oh My Zsh. This plugin changes the colors of the `zsh` command you just entered. What may seem superfluous at first glance is actually a great tool for recognizing before you press Enter for the command whether it contains a typo (the command is then displayed in red instead of green) or whether a quotation mark is not closed.

For the installation, you need to run the following command, which you can also find on the GitHub project page (see *https://github.com/zsh-users/zsh-syntax-highlighting/blob/master/INSTALL.md*):

```
user$ git clone https://github.com/zsh-users/zsh-syntax-highlighting.git \
  ${ZSH_CUSTOM:-~/.oh-my-zsh/custom}/plugins/zsh-syntax-highlighting
```

After that, you just need to add `zsh-syntax-highlighting` to the `plugin` line of `.zshrc`.

Chapter 9
Files and Directories

This chapter describes the handling of files. In detail, we'll cover the following topics:

- Files, directories, and links
- Copying, moving, deleting, and finding files
- Access rights for files (including access control lists [ACLs])
- Linux directory structure
- Device files

File system administration

In a way, the continuation to this chapter follows in Chapter 18. That chapter deals with questions that are primarily of interest to system administrators: What file systems are available? How do I access external data media and USB flash drives? How are file systems integrated into the directory tree (`/etc/fstab`, `mount options`)? How do I change the partitioning of a hard disk or SSD? How can a software RAID system be used? What is LVM? How can an entire file system be encrypted?

Backup

The topic of backups is covered in Chapter 28. There, I will introduce you not only to user interfaces for performing backups, but also to a wide range of commands for compressing files, bundling them into archives, synchronizing them, encrypting them, and so on.

9.1 Handling Files and Directories

Let me start with a brief summary of the most important facts about file names:

- On Linux, file names with lengths of up to 255 characters are permitted.
- A distinction is made between uppercase and lowercase letters.
- Spaces in file names are permitted but often cause problems when they are processed by scripts. File names with spaces or special characters must be placed in quotation marks, such as `"a b"`. Avoid using spaces!
- International characters in the file name are allowed, but can cause problems if different character sets are used, such as on a network. All common Linux distributions use Unicode UTF-8 as their default character set. From the point of view of the Linux kernel, the file name is simply a byte sequence in which only the / character and the code 0 must not

occur. The question how this byte sequence is interpreted depends on the currently valid character set.

- File names may contain any number of dots. `README.bootutils.gz` is a normal file name indicating that it is a compressed readme file about boot utilities.
- Files that start with a dot are considered hidden files. Hidden files are usually not displayed by `ls` or by various file managers.

The size of files is almost unlimited with current Linux distributions and is in the terabyte range depending on the file system.

9.1.1 Directories

Directory tree

The directory tree of Linux starts in the root directory, */*. Drive specifications like `C:` are not possible on Linux. Within this book, all other directories are considered *subordinate*; figuratively speaking, the root directory is therefore right at the top. In some books, the naming system is reversed, which corresponds better to the tree image (root at the bottom, branching at the top), but does not correspond to common usage.

Linux beginners often find it difficult to find files in the widely ramified directory system. For a solution to this problem, read Section 9.3 and Section 9.8! There you will learn about various search tools and how the Linux directory tree is structured.

Home directory

In text consoles or terminal windows, the so-called home directory is initially automatically active. All files and subdirectories contained therein are yours. Other users with the exception of `root` are not allowed to modify or delete these files and, depending on the access rights settings, are not even allowed to read them.

The home directory is usually located at `/home/<loginname>/` in the Linux directory tree. Only for `root` is the home directory `/root`. As it would be cumbersome to always write out `/home/<loginname>`, your own home directory can be abbreviated by using the tilde character, `~` (see Table 1.1). For accessing the home directories of other users, the notation `~<loginname>` is also possible.

The . and .. directories

In each directory, there are two special subdirectories, which are used for the formal management of the directory hierarchy. The directory named `.` is a reference to the current directory, while the `..` directory is a reference to the parent directory. The following two copy commands show how you can use these directories. The first command copies the `/etc/fstab` file to the current directory, and if the current directory is `/home/name`, then the new file is named `/home/name/fstab`:

```
user$ cp /etc/fstab .
```

The second example first activates the `linuxbook` directory in the home directory by using `cd`. The copy command `cp` then creates a backup copy of the `fileuse.tex` file, which contains the text of this chapter. The name of the backup copy is `~/fileuse.tex.bak`:

```
user$ cd ~/linuxbook
user$ cp fileuse.tex ../fileuse.tex.bak
```

So if the home directory is `/home/name`, then you've just created a copy of `/home/name/linuxbook/fileuse.tex`. The full name of the backup copy is `/home/name/fileuse.tex.bak`. More `cp` examples follow in the next section.

Character	Meaning
`~`	Home directory
`.`	Current directory
`..`	Parent directory of the current directory

Table 9.1 Short Notations for Important Directories

9.1.2 Elementary Commands for Editing Files and Directories

As an alternative to the desktop system's file manager, experienced Linux users like to use text-based commands to manage files and directories. Table 9.2 summarizes the most important commands.

Command	Function
`cd`	Changes the current directory
`cp`	Copies files
`j`	Changes to a directory that was last used
`less`	Displays text files page by page
`ls`	Displays all files in a directory
`mkdir`	Creates a new directory
`mv`	Moves files or changes their names
`rm`	Deletes files
`rmdir`	Deletes directories

Table 9.2 Elementary Commands for Handling Files and Directories

Changing directories using cd

The `cd` command changes to another directory. `cd -` changes back to the last active directory, `cd ...` changes to the parent directory, and `cd` without further parameters changes to the home directory:

```
user$ cd /etc/samba
```

Changing directories using j

In some distributions, you can also use the `j` command from the `autojump` package to change directories. As a rule, you must install this package first. Autojump remembers which directories you work in most often. `jumpstats` provides you with the corresponding statistical data if needed.

If you want to change to a previously used directory, you need to run `j abc`, where `abc` represents the first letters of the directory name. Entering the often long path to the directory is not necessary!

If there are multiple matching directories, `j` will change to the directory you used most recently. In addition, you can use the Tab key to choose between the directories available for selection.

The autojump website describes `j` as a `cd` command that learns along, which is quite an apt description. It only takes a short time to get used to `j`; after that, you don't want to miss the command (see *https://github.com/joelthelion/autojump/wiki*). `j` cannot replace `cd`, however, but only supplement it: Completing the pathname by using the Tab key works only for directories that are already in the autojump database. Thus, if you type `cd /e` Tab, `/e` is usually added to `/etc/`. The analogous completion of `j /e` Tab, on the other hand, doesn't work until `/etc` is already in the autojump database. In other words, to change to a directory autojump doesn't know, it's still better to use `cd`. (You can also get to the relevant directory by using `j`, but you have to type the entire directory name). Too bad: it would be even more elegant if `j` could be used as a full `cd` replacement!

Listing files

`ls` returns a list of all files in the current directory. If you want to see hidden files also, you must specify the `-a` option as well. If you are interested not only in the file name but also in the file size, owner, and other details, the `-l` option will help you. By default, the output of `ls` is in alphabetical order. To sort the file list by the time of the last modification, file size, or file identifier, you want to use the `-t`, `-S`, or `-X` options. `-r` reverses the sort order. The following command shows all `*.tex` files in the `linuxbook` directory, ordered by size (the largest file first):

```
user$ ls -l -S linuxbook/*.tex
...
-rw-r--r-- 1 kofler kofler  30113 2023-05-11 09:09 linuxbook/intro.tex
-rw-r--r-- 1 kofler kofler  63173 2023-01-29 08:05 linuxbook/kde.tex
```

```
-rw-r--r-- 1 kofler kofler  76498 2023-06-08 15:43 linuxbook/kernel.tex
...
```

Briefly, some comments on the interpretation of the `ls` results: the 10 characters at the beginning of the line indicate the file type and the access bits. The following file types are possible: the hyphen `-` for a normal file, `d` for a *directory*, `b` or `c` for a device file (*block* or *char*), or `l` for a symbolic link. The next three characters (`rwx`) indicate whether the owner is allowed to read, write, and/or execute the file. Analogous information follows for the members of the group as well as for all other system users.

The number following the 10 type and access characters indicates how many hard links reference the file. Section 9.2 describes what links are. Details on managing access to Linux files follow in Section 9.5. The other columns indicate the owner and group of the file (here `kofler` in each case), the size of the file, the date and time of the last modification, and finally the file name.

ls colors

In most distributions, `ls` is configured to display files and directories in different colors depending on their type. If that isn't the case in your distribution, you can achieve this effect via the `--color` option. Conversely, to prevent the colored display, you must run `l --color=none`.

Recursive ls

`ls` takes into account only the files of the current directory. To include the files from subdirectories as well, you want to use the `-R` option. By the way, this option is also available for many other commands. The following command lists all files in all subdirectories. This list usually becomes quite long. For this reason, `| less` passes the result of `ls` on to `less` so that you can scroll through the result:

```
user$ ls -lR | less
```

Copying files

`cp name1 name2` copies the `name1` file. The copy gets the name `name2`. To copy multiple files, you need to call the command in the following way: `cp name1 name2 ... target directory`. The following commands make `linuxbook` the active directory, create `bak` as a subdirectory, and copy all `*.tex` files there:

```
user$ cd linuxbook
user$ mkdir bak
user$ cp *.tex bak/
```

Copying directories

To copy entire directories along with their contents, you must use `cp -r`. The `-r` option causes the entire contents of the source directory to be processed recursively (including hidden files). If you want access rights and times to be preserved when copying, you need to use the `-a` option instead of `-r`.

The question whether the source directory itself or only its content is copied is somewhat difficult. If the target directory already exists, the new subdirectory named `sourcedirectory` will be created in it and the entire contents of the source directory will be copied there. If, on the other hand, the target directory does not yet exist, it will be created. In this case, only the *contents* of the source directory will be copied to the newly created target directory, but not the source directory itself. The following examples illustrate the difference:

```
user$ mkdir test
user$ touch test/a
user$ mkdir test/b
user$ touch test/b/c
user$ mkdir target1
user$ cp -r test target1          (the target1/ directory already exists)
user$ ls target1
test
user$ cp -r test target2          (the target2/ directory does not exist yet)
user$ ls target2
a  b
```

Deleting files and directories

`rm file` deletes the specified file irrevocably. Usually, `rm` can only be used for files, but not for directories. For directories, the `rmdir directory` command is provided, but it works only if the directory is empty. In practice, you will often use `rm` with the `-rf` option to delete directories. This means that recursively all subdirectories and files are deleted without confirmation:

```
user$ rm -rf linuxbuch-bak/       (delete backup directory)
```

It should therefore be clear to you that `rm -rf` is a very dangerous command!

9.1.3 Determining Space Requirements of Files and Directories

Although `ls -l` tells you how big a file is, you'll often want to know how much space the files take up in the entire directory, how much free space is still available on the hard disk, and so on. The `df` and `du` commands help you in this regard.

Determining free hard disk capacity

`df` shows for all partitions or disks mounted in the file system how much storage space is available in total and how much of it is still free.

In the following example, `df` returns results for four partitions or volumes. The `-h` option causes all capacity information to be specified in readable numbers in KiB, MiB, or GiB. `df` normally also displays various temporary

file systems that are only used for internal administration and are not intended for storing regular files. You can avoid including those file systems in the `df` result by using the `-x tmpfs` and `-x squashfs` options (the latter option eliminates the Ubuntu-specific snap file systems):

```
user$ df -h -x tmpfs -x squashfs
File system          Size    Used Avail. Used% Mounted on
udev                  16G       0    16G    0% /dev
/dev/nvme0n1p4       119G     17G    96G   16% /
/dev/nvme0n1p1      1021M    238M   784M   24% /boot/efi
/dev/mapper/vg-root  1,5T    223G   1,2T   16% /crypt
```

You can also use `df` to determine which partition a directory is physically located in. In the following example, the `/crypt/home/kofler` directory is located in a logical volume that is responsible for the entire `/crypt` directory. On my notebook computer, there is an encrypted file system in this location:

```
user$ df -h /crypt/home/kofler/
/dev/mapper/vg-root  1.5T    223G  1.2T   16% /crypt
```

Determining the directory size

`du` determines the space requirements for the current directory as well as for all subdirectories contained therein. The `-h` option again makes sure that the result gets displayed in readable form (not in KiB blocks):

```
user$ du -h photos/2023
74M     photos/2023/04-easter
66M     photos/2023/06-miscellaneous
162M    photos/2023/08-corsica
...
2.0G    photos/2023
```

`du` does not provide the option to sort the result. This is exactly what the interactive `ncdu` command can do, which you have to install separately. It also provides the option to navigate through subdirectories using the cursor keys. I highly recommend it!

9.1.4 Wildcard Characters

In your daily work with files, you will often edit whole groups of files—for example, all files with the `.pdf` extension. To make this possible, wildcard characters are provided for entering Linux commands (see Table 1.3). Behind the scenes, the globbing shell mechanism is responsible for parsing these characters (see Chapter 7, Section 7.6).

Character	Meaning
?	Exactly one character of your choice
*	Any number (also zero) of any characters
[abc]	Exactly one of the specified characters
[a-f]	A character from the specified range
[!abc] or [^abc]	None of the specified characters

Table 9.3 Wildcard Characters for Filenames

*** and ?** ? is used to specify *any* character, and * is used to specify any number of characters (including zero). Those who are still familiar with DOS will not recognize any difference at first glance. But this impression is not quite true:

- * captures almost all characters, including dots, so long as they are not at the beginning of the file name. If you want to edit all files, then you must use * on Linux, not *.*! Notes on hidden files follow later in this chapter.
- Even multiple wildcard characters do not throw Linux off balance. For example, you can use *graph* to search for all files that contain graph in their name, such as graphic.doc, applegraph, and README.graph.

[ ] and [!] If the ? and * wildcard characters are too general for you, you can achieve a higher degree of restriction by specifying square brackets. [abc] is a placeholder for one of the three letters a, b, or c. If a hyphen is specified between two letters or digits within the square brackets, then it represents a character between them:

- [a-f]* thus covers all files that begin with a letter between a and f.
- *[_.-]* represents all files that contain at least one dot, underscore, or hyphen somewhere in their file names.
- The expression can be negated by an exclamation mark. Thus, [!a-z]* refers to all files that start with an uppercase letter or with a special character.
- *.[hc] captures all files ending with .c or .h.

*** for directories** The wildcard characters can also be used for directories. */*.jpg captures all *.jpg files located in subdirectories of the current directory (only one level below, so not also files in sub-sub-directories). /usr/*bin/* captures all files in the /usr/bin and /usr/sbin directories.

The shell from which the command is called is responsible for analyzing the wildcard characters, not the command called in each case. bash, the most

commonly used shell on Linux, uses a lot of other special characters besides the wildcard characters just described, which have a special effect when a command is executed (see Chapter 7, Section 7.6).

Example

The following command copies all `*.c` files from the `project` directory to the current directory:

```
user$ cp project/*.c .
```

9.1.5 Complications with Using Wildcard Characters

Dealing with wildcard characters looks easier at first glance than it actually is. If you have difficulties with wildcard characters, you should just do some experiments with `echo <wildcard_character>`. This command simply displays all file names captured by a wildcard character combination on the screen without changing the file names.

One problem is that `*` captures not only files, but also directories. For this reason, `ls *` displays not only all files in the current directory, but also the contents of all subdirectories captured via `*`. With the `ls` command, this problem can be circumvented by using the `-d` option; however, this option is not available with other commands.

Editing directories using */.

If you only want to edit directories, but no files, the character combination `*/.` will help you: it is only true if the searched object (`*`) contains a back reference to itself (`/.`). This condition applies only to directories:

```
user$  echo */.
```

Problems with *.extension

The fact that not the respective program but the shell is responsible for the processing of the wildcard characters also has some drawbacks. For example, it's impossible to search for `*.tex` files even in subdirectories by using `ls -R *.tex`. The `-R` option for the `ls` command actually causes a recursive search of subdirectories.

The reason that the command still doesn't work is simple: the shell expands the `*.tex` pattern for the *current* directory and passes the list of found files to `ls`. This command then displays information about these files. If you don't have any directories with the extension `.tex`, `ls` is finished with that; even the `-R` option can't change that anymore. Only directories that are passed as parameters are searched recursively.

To search for files, Linux therefore provides the much more flexible `find` command. The following example shows a list of all `*.tex` files in the current directory and in all subordinate directories (basic information on `find` and additional examples follow in Section 9.3.3):

```
user$  find . -name '*.tex'
```

Renaming files

Unfortunately, in Linux you cannot rename all `*.x` files to `*.y` files using the `mv *.x *.y` command. The reason for this restriction is again the same as described earlier: the shell replaces `*.x` with the list of all files matching this pattern; for `*.y`, there may be no matching files at all. For this reason, a list of several files and the `*.y` expression are passed to the `mv` command—and `mv` doesn't know what to do with these arguments.

Here's a concrete example. Let's suppose that the current directory contains only the following files: `marcus.x`, `peter.x`, and `janet.x`. When you run `mv *.x *.y`, the shell replaces the `*.x` pattern with the three named files. The shell doesn't find any matching files for `*.y` and passes the pattern as is. Only now will the `mv` command be started. It gets the following parameters, with which it can do nothing, as expected:

```
user$ mv marcus.x peter.x janet.x *.y
```

Even if `marcus.x peter.x janet.x marcus.y peter.y janet.y` were passed to `mv` as a parameter list, the effect would not be the desired one. `mv` is generally not able to rename multiple files. So either *multiple* files are moved to another directory or only *one* file is renamed.

Renaming using sed and for

Of course, Unix experts have found a solution to this problem as well: You must use the `sed` stream editor. Because of the rather complicated operation of `sed`, examples like the following are really only suitable for shell programming.

Here's briefly how it works: `ls` returns the list of files to be renamed and passes it to `sed`. `sed` uses the `s` command (regular search and replace) to create a list of `cp` commands from it and in turn passes it to a new `sh` shell, which finally executes the commands. The following line copies all `*.x` files into `*.y` files:

```
user$ ls *.x | sed 's/\(.*\)\.x$/cp & \1.y/' | sh
```

Another solution is to formulate a small loop. The following command creates backup copies of all `*.tex` files with the `tex~` extension. The `~` suffix is often used to indicate backup copies:

```
user$ for i in *.tex; do cp $i $i~; done
```

9.1.6 Hidden Files and Directories

On macOS and Linux, files and directories whose names begin with a dot are considered hidden. The `*` wildcard character doesn't even consider all files in a directory. Files starting with a dot (often configuration files that should be invisible) are ignored.

Now, if you think you can capture invisible files using `.*`, things will get worse: this refers not only to invisible files that start with `.`, but also to the `.` and `..` directories (i.e., the current and the parent directory). If the respective command is able to process entire directories, the consequences can be fatal.

The problem can be avoided by using the search pattern `.[!.]*`, which covers all file names for which the first character is a dot, that have at least one other character that is not a dot, and that have any number (including zero) of other characters:

```
user$ echo .[!.]*
```

The `-a` option can be used along with the `ls` command. It causes all files to be displayed, even invisible ones. However, masks (such as `*rc*`) must not be specified when using `ls` in this way. `-a` only works if `ls` is allowed to search for the files itself and the shell doesn't take over this task.

In this case too, only `find` really works universally. The following command finds all hidden files in the current directory:

```
user$ find -maxdepth 1 -type f -name '.*'
```

9.1.7 Special Types of Files (Links, Devices, and the Like)

In addition to ordinary files, Linux knows a number of special types, like directories, links, and device files for accessing hardware components. `ls -l` identifies such special types by the first character of the output (see Table 9.4):

```
user$ ls -l /dev/s*
brw-rw----. 1 root disk ... /dev/sda                    (block device)
brw-rw----. 1 root disk ... /dev/sda1                   (block device)
crw-rw----. 1 root disk ... /dev/sg0                    (character device)
lrwxrwxrwx. 1 root root ... /dev/stderr -> /proc/self/fd/2  (link)
...
```

Character	Meaning
-	Ordinary file
d	Directory
l	Symbolic link
c	Character-based device (character device)

Table 9.4 Identification of Special Files in First Column of ls Results

Character	Meaning
b	Block-based device (block device)
s	Socket (interprocess communication)
p	Pipe, first in first out (FIFO)

Table 9.4 Identification of Special Files in First Column of ls Results (Cont.)

You'll learn more about links in the next section, and about device files in Section 9.9. FIFO files allow one process to write to the file and a second process to read the data from it. Socket files are also used for communication between processes, but they are implemented much more efficiently internally.

9.2 Links

Hard links

Links are references to files or directories. They allow you to access the same file from different locations in the directory structure without having to physically store that file multiple times. Links are thus an important tool for avoiding redundancies. In the Linux file system, links are particularly common in `/bin` and `/lib` directories. For example, take a look at the result of the `ls -l /usr/bin` command.

The easiest way to understand links is to use an example. Let's suppose there is a file named `abc` in the `test` directory; the `ln abc xyz` command apparently creates a new file named `xyz`. But in reality, `abc` and `xyz` are just two references to one and the same file. The only way to check this is to use the `ls` command using the `-l` option. In the second column, it indicates how many links reference a specific file; in this example, that's two files. If the `-i` option is also used, `ls` also specifies the inode of the file, which is identical for links. *Inodes* are internal identification numbers of the file system.

```
user$ ls -li
 59293 -rw-r--r--   1 root     root    1004 Oct  4 16:40 abc
user$ ln abc xyz
user$ ls -li
 59293 -rw-r--r--   2 root     root    1004 Oct  4 16:40 abc
 59293 -rw-r--r--   2 root     root    1004 Oct  4 16:40 xyz
```

If you now change one of the two files (no matter which one), the other file will automatically change as well—because in reality there is only one file! If you delete one of the two files, you will only reduce the number of links.

Backup Files from Linked Files Are an Issue

If you edit hard-linked files using a text editor, strange results sometimes occur: the link references the backup file after the first save and nothing on the second save.

Here's why: Some editors create a backup file when saving by renaming the existing file, like `abc` to `abc~`. The changed file is created completely new, gets a new inode, and is thus free of any links. A solution to this problem can be to use symbolic links discussed ahead.

Symbolic links

Linux uses two types of links. The previous example presented hard links as created by default by the `ln` command. Aside from that, if `ln` is used with the `-s` option, the command creates symbolic links. Symbolic links are sometimes also referred to as *soft links*. They have the advantages that they can point from one disk to another and that they can be applied not only to files but also to directories. Neither is usually possible with hard links. A special case is hard links to directories, which are possible but can only be created by `root`.

With symbolic links, `ls` indicates where the original file is located. However, no counter is maintained to indicate from how many places the source file is referenced.

Internally, the difference between hard links and symbolic links is that in the one case the inode is saved, and in the other case the file name or (for links beyond a directory) the path specification is saved:

```
user$ ln -s abc efg
user$ ls -li
 59293 -rw-r--r--   2 root     root    1004 Oct  4 16:40 abc
 59310 lrwxrwxrwx   1 root     root       3 Oct  4 16:52 efg -> abc
 59293 -rw-r--r--   2 root     root    1004 Oct  4 16:40 xyz
```

Tip

Before setting up a symbolic link, you should always change to the directory that will contain the link. Otherwise, the link may not point where you expect it to.

Symbolic links behave a little differently than fixed links. Deleting the source file (i.e., `abc` from the previous example) does not change the link to this file, but `efg` then references a file that does not exist at all. If, on the other hand, the symbolic link is deleted, this will have no effect on the source file.

Symbolic links can be created not only for files, but also for directories. This can cause some confusion because a symbolic link apparently duplicates entire directory trees. But in reality the directory link is just an additional path to the same files and subdirectories.

In general, you should try to use only relative paths in links if possible. This will help you to avoid problems that can arise when you mount directories via NFS or when you move directories.

Hard Links versus Symbolic Links

Both hard links and symbolic links have advantages. Symbolic links are easier to use, but hard links consume less memory and are faster.

9.3 Finding Files (find, grep, and locate Commands)

Linux provides various options to search for files (see Table 9.5). The most suitable command depends on the type of file (text file, program, etc.) and what information is known—such as the parts of the file name or search terms for the content.

Command	Function
`grep`	Searches for text in a text file
`find`	Searches for files by name, date, size, and so on
`locate`	Searches for files by their name
`whereis`	Searches for files in predefined directories
`which`	Searches for programs in `PATH` directories

Table 9.5 Commands for Finding Files

9.3.1 which and whereis

`which` searches for the specified command. It returns the full name of the command that would be executed if the command name were called without path information. `which` searches only the directories specified in `PATH` and therefore works exceptionally fast. `PATH` contains a list of directories where programs are located. Note, however, that `PATH` contains more directories for `root` than for ordinary users. So if you are looking for system commands, you have to log in as `root`:

```
user$ which emacs
/usr/bin/emacs
```

whereis searches all common paths for binary files, configuration files, man pages and source code for the specified file name. whereis thus covers more directories than which and is not limited to programs. However, it fails for files that are not in the directories predefined for whereis (see man whereis):

```
user$ whereis fstab
fstab: /etc/fstab /usr/include/fstab.h /usr/share/man/man5/
fstab.5.gz
```

9.3.2 locate

locate <pattern> finds files where the specified search pattern occurs in the full file name—that is, in the path plus the file name.

The search is very fast: locate does not search the file system, but accesses a database that contains a list of all file names in the file system. Depending on the distribution, locate displays only those files to which the user actually has access. If necessary, you can run locate as root when searching for system files. locate can only be used if the appropriate package is installed, which is not the case by default for all distributions.

Examples

The following command searches the X configuration file, xorg.conf:

```
user$ locate xorg.conf
/usr/share/X11/xorg.conf.d
/usr/share/X11/xorg.conf.d/10-evdev.conf
/usr/share/X11/xorg.conf.d/10-quirks.conf
/usr/share/X11/xorg.conf.d/11-evdev-quirks.conf
...
```

The search for dvips returns a lot of hits (if this package and LaTeX is installed) because the search term occurs in multiple directory names. Instead of displaying all search results, they are counted using wc:

```
user$ locate dvips | wc -l
    421
```

The number of results will be much smaller if you search only for files ending with dvips:

```
user$ locate '*dvips'
/usr/bin/dvips
/usr/bin/odvips
/usr/bin/opdvips
```

```
/usr/bin/pdvips
/usr/local/texmf/dvips
/usr/local/texmf/fonts/map/dvips
...
```

updatedb

The quality of the search results is highly dependent on the up-to-dateness of the database for `locate`. In most distributions, the `locate` database is updated once a day by the `updatedb` command. Of course, `updatedb` can also be run manually at any time, but this requires `root` privileges.

Distribution-specific details

For performance reasons, `locate` and `updatedb` are being installed less and less by default. This can be solved in most distributions by installing the `plocate` package. The file database is located in the `/var/lib/plocate/#plocate.db` file and is regularly updated by a cron or timer job. The `/etc/updatedb.conf` configuration file determines which directories and file systems are not included (e.g., network file systems).

On openSUSE, `locate` is implemented by the `mlocate` package, which must be installed if required. The configuration is also done in the `/etc/updatedb.conf` file. The resulting database ends up in the `/var/lib/mlocate` directory.

9.3.3 find and grep

`find` is a powerful and complex command for searching files. It takes into account various search criteria: a pattern for the file name, the file size, the date of creation or last access, and so on. A complete reference of all options is contained in `man find`.

However, the following examples are probably the best introduction to using `find`. Note that `find` is a comparatively slow command because it searches the file system directory by directory.

find

Without any additional parameters, `find` returns a list of all files in the current directory and in all subdirectories:

```
user$ find
...
```

The following command searches for all files in the current directory and in all subdirectories that begin with `.e`:

```
user$ find -name '.e*'
./.evolution
./.emacs
```

```
./.emacs~
./.esd_auth
...
```

Starting from the /usr/share/texmf directory, find searches for all *.tex files in a directory ending with latex:

```
user$ find /usr/share/texmf -path '*latex/*.tex'
/usr/share/texmf/ptex/platex/base/plnews03.tex
/usr/share/texmf/ptex/platex/base/kinsoku.tex
...
```

In the next example, find searches for all directories inside /etc/. Ordinary files in /etc, on the other hand, will not be displayed. The results list is sorted alphabetically by sort, which is not the case by default:

```
root# find /etc -type d | sort
/etc
/etc/acpi
/etc/acpi/actions
...
```

Next, find searches for all files in the (sub)directories of /home that belong to users of the users group and whose contents have been modified in some way in the last five days:

```
root# find /home -group users -mtime -5
...
```

find -mtime +5 finds files that were modified *more* than five days ago, and -mtime 5 returns such files that were modified *exactly* five days ago. find calculates in multiples of 24 hours from the current time. If you use the -ctime option instead of -mtime, the *inode change time* will be taken as the time of change. This point in time also changes, for example, if the access rights are changed rather than the content.

The following command deletes all backup files in the current directory and in all subdirectories. The list of all relevant files is created using find and passed on to rm by means of command substitution $(command):

```
user$ rm $(find . -name '*~')
```

If there are *many* files, an error will occur when you run the above command: The command line with all *~ files will become so long that it exceeds the maximum command line length. In such cases, you must use either the -exec option of the find command or the xargs command.

grep

The grep command searches a text file for a search pattern. Depending on the options settings, the command then displays the text passages it has found or simply indicates in how many lines the search pattern was found. The search pattern is a so-called regular expression. The following command searches all *.tex files in the current directory for the string "emacs". The list of all lines found, each preceded by the file name, is displayed in the terminal.

```
user$ grep emacs *.tex
...
```

Here, grep determines how often the arctan function is used in the specified *.c files:

```
user$ grep -c 'arctan\(.*\)' *.c
```

grep -v returns all lines that do not contain the search pattern. In the following example, grep removes all lines from configfile that begin with the # character—that is, all comments. In addition, the trailing cat command eliminates all empty lines. The final result will be stored in the nocomments file. The instruction is useful when a few configuration lines are drowned out by hundreds or thousands of comment lines:

```
user$ grep -v '^#' configfile | cat -s > nocomments
```

Combining find and grep

You can also combine find and grep to perform particularly effective searches. In the following example, find scans all *.tex files to see if they contain the "emacs" string. If they do, the respective file name will be displayed on the screen. Note that the -print option must not be specified before -exec. Unlike the grep emacs *.tex example, this example also takes into account *.tex files in subdirectories nested to any depth:

```
user$ find -name '*.tex' -type f -exec grep -q emacs {}  \; -print
...
```

The following command searches all files in the current directory that are smaller than 10 KiB for the regular expression, case.*in. The list of found files is saved in the result file. By limiting the file size to 10 KiB, an attempt is made to exclude binary files, which are usually considerably larger, from the search:

```
user$ find -name '*' -maxdepth 1 -size -10k -exec grep -q \
        case.*in {} \; -print > result
```

9.4 Greater Convenience with Modern Commands

Most of the Linux commands I present in this book are decades old. That's not necessarily a bad thing: the commands are established, mature, and available on virtually any Linux machine or cloud instance.

Nevertheless, more and more "new" commands have been introduced in recent years that perform similar functions to existing commands but are more convenient to use and present their results more clearly and in colors (see Table 9.6). Most of these commands are explicitly intended for interactive use, not for use in scripts!

exa/eza

I would like to briefly introduce you to four selected tools here. The command you probably run most often in the terminal is `ls`. The `exa` command or its fork `eza` (*https://github.com/eza-community/eza*) performs the same task, but formats the result in a more readable way and has various additional functions. You can try, for example, `exa -l --tree` or `exa -l --git`! `exa` or `eza` can be installed as a package in many distributions. (For RHEL, you must enable the EPEL package source.)

bat

In the list of the most important commands, the `cat` and `less` commands follow immediately after `ls` to view the contents of a text file. How many times have you been annoyed by the monotonous presentation of the text? This can be remedied by the `bat` command, which is available as a package in most distributions. It tries to recognize the content of the text and then uses appropriate syntax highlighting and displays line numbers (see Figure 9.1).

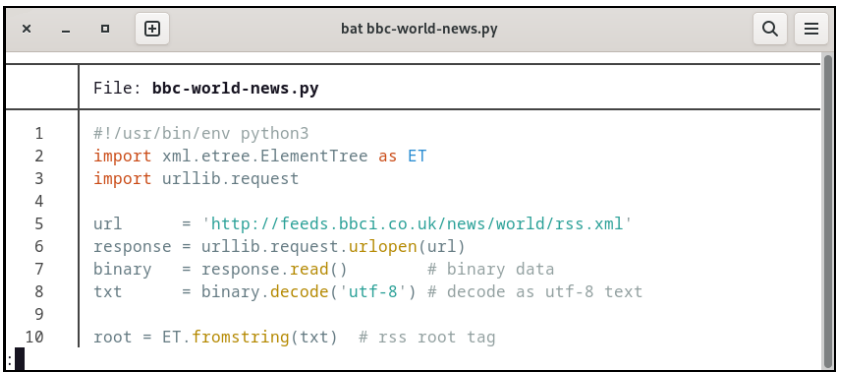


Figure 9.1 Representation of Python Script in Terminal Using bat

`bat` relies on your terminal to use a dark background by default. To make the screenshots in this book easier to read, I use a light background. In this case, you must pass an appropriate theme option to `bat` (such as `--theme`

gruvbox-light) or set it permanently in the .config/bat/config configuration file.

fzf fzf stands for *fuzzy finder*. If the command is started without any other options, it searches the current directory like find . -type f for files and displays the result list. By entering a search pattern, you can filter the result interactively and then select a file using the cursor keys and Enter or multiple files by pressing the Tab key. The file names will then be displayed.

This basic function sounds rather unspectacular. As a matter of fact, however, fzf provides countless usage options. For example, you can use xargs to pass the selected file to another program:

```
user$ fzf | xargs -r $EDITOR
```

Provided that fzf is installed correctly, fzf can also be triggered by typing ** and pressing Tab, allowing the interactive completion of commands: Using emacs ** Tab, you can open the previously selected file. When you enter kill -9 ** and press Tab, fzf provides processes you can select, and when you enter ssh ** and press Tab, it provides host names for selection. Ctrl+T lists all files, Ctrl+R lists recently executed commands, and Alt+C helps fzf to change to a subdirectory.

> **Tip**
>
> On YouTube, you can find various videos that illustrate the application of fzf pretty well. Even though I'm not usually a video fan, in this case, it's worth it!

ncdu If you want to know how much space is required by various directories and subdirectories, you usually use du. However, the result is not ordered by size. A much more convenient option is ncdu, which is available in the package of the same name in many distributions. ncdu first runs through the entire directory tree, which can take a while. Then it shows only the first directory level, with the largest directory first. Subsequently, you can comfortably explore the subdirectories using the cursor keys.

Command	Replaces.../Similar Function To...
bat	cat/less with syntax highlighting and line numbers
duf	df

Table 9.6 New Commands for Standard Tasks

Command	Replaces.../Similar Function To...
erd (erdtree)	du, ls, tree
fd	find
fx	less for JSON files
fzf	Fuzzy finder (find, grep)
glow	less for markdown files
j	cd
lfs	df
lsd	ls
ncdu	du, but interactively
tldr	man (example-based short version)
z	cd

Table 9.6 New Commands for Standard Tasks (Cont.)

Disadvantages of the New Commands

The main problem with the commands presented here is their installation. If you work primarily on *one* Linux computer, the required installation commands aren't important, especially since many of the tools presented here are already so well established that there are packages in the usual package sources. But in my day-to-day work, I'm constantly switching back and forth between countless Linux installations, most of which are in virtual machines, in the cloud, or in Docker containers. Before I go to the trouble of installing duf, I use the conventional df.

9.5 Access Rights, Users, and Group Membership

Linux is a multiuser system and therefore needs mechanisms to control who can access which files, who can modify them, and so on. The basis of the access system is the management of users and groups, which is described in Chapter 14, Section 14.5. For now, let's just assume that every user on Linux is assigned to a user account and one or several groups. This minimum requirement is sufficient to understand the concept of file access rights.

9.5.1 Access Rights for Files

Access rights per file

The following information is stored with each file or directory:

- The owner of the file
- A group to which the file is to be assigned
- Nine access bits (`rwxrwxrwx` for read/write/execute for the owner, for all group members and for the rest of the world)
- A few more additional bits for special functions

The owner of a file is usually the person who created the file. The primary group of the owner is typically used as the group—as the owner's default group.

The `r`, `w`, and `x` access information controls who may *read*, *write*, and *execute* the file. This information is stored separately for the owner, for the group, and for all other users. This makes it possible to give the owner more rights than other users. The information is usually referred to as *access bits* because it is stored internally as a number with bitwise coding.

Who Is Allowed to Delete a File?

The access rights to a file have *no* influence on who is allowed to delete a file. This is decided solely by the person who has access to the *directory* in which the file is located! A file may be deleted by anyone who has the `w` and `x` rights for the directory. More information about directory access rights follows in the next section.

Interpreting the ls output

The access bits, the owner, and the group membership of a file can be viewed via `ls -l`. For a typical text file, `ls` returns the following result:

```
michael$ ls -l file.txt
-rw-r-----   1 michael  users        3529 Oct  4 15:43 file.txt
```

Brief interpretation: The first character indicates the file type. The `-` sign means that it is a normal file. Other possibilities are `d` for a *directory*, `l` for a symbolic link, and so on.

The three characters `rw-` indicate that the file can be read and modified by the owner `michael`. For a text file, the first `x` bit is disabled, so the file cannot be executed.

The following three characters, `r--`, indicate that all members of the `users` group may read the file but not modify it.

The final three characters, `---`, indicate that other users—who are neither `michael` nor members of the `users` group—are not allowed to read or modify the file.

chmod and chgrp

If `michael` wants this file to be readable by all users, then he must enable the final `r` bit. To do this, he uses the `chmod o+r` command:

```
michael$ chmod o+r file.txt
michael$ ls file.txt -l
-rw-r--r--   1 michael  users        3529 Oct  4 15:43 file.txt
```

Sometimes *two* or more users should be allowed to modify the file. A new group to which these users belong can be created for this purpose. If `michael` and `catherine` form the documentation team of a company, the group name would be something like `docuteam`. Then the group membership can be changed using `chgrp`:

```
michael$ chgrp docuteam file.txt
michael$ chmod g+rw file.txt
michael$ ls file.txt -l
-rw-rw-r--   1 michael  docuteam     3529 Oct  4 15:43 file.txt
```

As a matter of fact, file sharing is even a little more tricky: it is also necessary to ensure that all users have access to the *directory* in which the files are located. More details on this topic will follow shortly.

Octal notation

Instead of the `rwxrwxrwx` notation, the nine access bits as well as three additional special bits are often also represented in octal. This is probably the most popular application of this number system based on the number eight to date. The access bits for the user, the group, and all others are each assigned a digit (see Table 9.7).

Code	Meaning
4000 = s = setuid 2000 = s = setgid 1000 = t = sticky	Special bits
400 = r = read 200 = w = write 100 = x = execute	Access bits for the owner (u = user in `chmod`)
40 = r 20 = w 10 = x	Access bits for group members (g = group)

Table 9.7 Octal Codes for Access Bits

Code	Meaning
4 = r 2 = w 1 = x	Access bits for all others (o = others)

Table 9.7 Octal Codes for Access Bits (Cont.)

Each digit is composed of the values 4, 2, and 1 for r, w, and x. 660 thus means rw-rw----, while 777 stands for rwxrwxrwx. The setuid, setgid, and sticky bits presented in Section 9.6 have octal values of 4000, 2000, and 1000, respectively.

The chmod command enables you to also set the access bits in octal, which many experienced users prefer because of the reduced typing effort:

```
user$ chmod 640 file.txt
```

Surprisingly, however, ls is not able to represent the access bits in octal form. The stat command can help you in this respect:

```
user$ stat -c "%a %n" *
755 php53 samples
550 Private
755 samples
...
```

Access rights to devices

Access to hardware components such as hard disks, DVD drives, interfaces, and so on is provided on Linux via devices (see Section 9.9). To be able to specifically control which user may access which devices, different user groups are assigned to the devices. For example, the /dev/ttyS* devices for the serial ports are assigned to the dialout group on Debian and Ubuntu:

```
root# ls -l /dev/ttyS1
crw-rw----   1 root     dialout       5,  65 Jul 18  /dev/ttyS1
```

If the system administrator wants the user herbert to be allowed to use the serial port, he adds herbert to the dialout group:

```
root# usermod -a -G dialout herbert
```

9.5.2 Access Rights for Directories

r, w, and x access to directories

Basically, the nine access bits are also valid for directories, but they have a slightly different meaning there:

- The r bit allows you to get the list of file names (ls command).

- The `w` bit gives you the right to change the contents of a directory—for example, to create a new file or rename or delete an existing file.
- The `x` bit allows you to change to a directory (`cd` command). However, you can only access files whose names you know. Only the `rx` combination makes it possible to process a directory correctly—for example, using `ls -l` to determine a list of all file names, including detailed information on each file. If both `x` and `w` are set, new files can be created in the directory.

The somewhat strange interpretation of the `r` and `x` access rights is due to the fact that directories are regarded by the file system as a special case of a file; the contents of the "file" directory is a listing of the names of the files that are in the directory and their inode numbers.

Determining who can do what

Table 9.8 summarizes which access rights are required for a directory and the file it contains in order to perform certain actions. The - sign in the File column indicates that the file access rights are not relevant. As usual, these rules apply only to ordinary users. `root` is allowed to do everything regardless of the access rights set!

Action	Command	File	Directory
Change to directory	`cd directory`	-	x
Determine list of files	`ls directory/*`	-	r
Read file information	`ls -l directory/*`	-	rx
Create new file	`touch directory/newfile`	-	wx
Read file	`less directory/file`	r	x
Change existing file	`cat >> directory/file`	w	x
Delete file	`rm directory/file`	-	wx
Run program	`directory/program`	x	x
Run script file	`directory/script`	rx	x

Table 9.8 Required Access Rights for Standard Actions

Nested directories

For nested directories, the `x` bit is particularly decisive for the base directories. If it isn't set, the subdirectories cannot be used. In practice, it's usually convenient to set the `r` bit for base directories as well. If the read permission is missing, the user must know the name of the subdirectory. The operations permitted in the subdirectory depend exclusively on the `rwx` bits of this directory. If the `rwx` rights are set for the subdirectory, files can be read,

created, modified, and deleted—even if the r and w rights are missing in the base directories!

For a better understanding, you can take a look at the / directory, the /home directory, and a user directory contained in it:

```
root# ls -ld /
drwxr-xr-x ... root root ... /home/
root# ls -ld /home/
drwxr-xr-x ... root root ... /home/
root# ls -ld /home/kofler/      (on Debian, Ubuntu)
drwxr-xr-x ... kofler kofler ... /home/kofler/
```

For / and /home, the following applies: Anyone is allowed to determine the names of the files and subdirectories contained in this directory, as well as query detailed information about them; that is, anyone can run ls -l /home. However, only root is allowed to set up new home directories using mkdir /home/newuser.

For /home/kofler, the following applies: Only the user kofler is allowed to create new files in this directory. All other users may look into the directory (ls -l), but they cannot change anything. It is up to the owner of the home directory to set the access rights for his or her own files and subdirectories, if necessary, so that read access is also impossible.

Note that kofler is of course allowed to create, modify, and delete files in the home directory, although this user has no write permissions in / and /home!

Some Linux distributions, such as Fedora or Red Hat, set the access rights for the home directories more restrictively and really only allow the owner to look into the directory:

```
root# ls -ld /home/kofler/      (on Fedora, RHEL)
drwx------ ... kofler kofler ... /home/kofler/
```

Sharing directories

The home directories just mentioned are an example of directories that are intended for a specific user only. But how do you set up directories that multiple users can share—for example, a project directory for users sebastian and matthew so that both can read *and* modify files in the directory?

The solution to such problems is the use of groups. You can create a new projectxy group and assign sebastian and matthew to this group. Then you still need a project directory that is assigned to this group:

```
root# addgroup projectxy
root# usermod -a -G projectxy sebastian
root# usermod -a -G projectxy matthew
```

```
root# mkdir -p /projects/xy
root# chgrp projectxy /projects/xy
root# chmod 755 /projects
root# chmod 2770 /projects/xy
root# ls -ld /projects/xy
drwxrws--- ... root projectxy ... /projects/xy
```

The setting of access rights for `/projects` is the same as for the `/home` directory: Only `root` is allowed to create new files and directories in it. All others may use the directories.

In the `/projects/xy` directory, all members of the `projectxy` group have read and write permissions, so in this example that's `sebastian` and `matthew`.

The only special feature is the setgid bit for this directory (octal code 2000). It causes files and directories created in this directory to be automatically assigned to the `projectxy` group, not to the group of the user who creates the file. This avoids the case where `sebastian` creates a new file, and `matthew` sees and reads it but cannot modify it. Background information on the setgid bit follows in the next section.

9.6 Special Bits and the umask Setting

You are now familiar with the principle of access rights, but two finishing touches are still missing: First, there are three more access bits with the strange names of setuid, setgid, and sticky, the meaning of which will be explained ahead. Second, there is still the question of who owns newly created files. Here the so-called umask setting plays a major role.

9.6.1 Setuid, Setgid, and Sticky Bit

The meaning of the three times three access bits, `rwxrwxrwx`, is easy to understand. In addition, three other pieces of information can be stored with the access information of files and directories: the setuid bit, the setgid bit, and the sticky bit. As a rule, only system administrators need to know these special bits.

Setuid bit (suid bit)

The *setuid bit* is often called the *suid bit* for short. It causes programs to always run as if the owner himself had started the program. Often the owner of programs is `root`; then anyone can run the program as if they were `root`. Internally, the user identification number of the owner of the file and not the UID of the current user is used for the execution of the program.

The bit is used to give additional rights to ordinary owners, which are valid only when this program is executed. The `/usr/bin/passwd` command is an

example of using the setuid bit. It allows each user to change their own password. However, the passwords are stored in the `/etc/shadow` file, to which only `root` has read and write access. For this reason, `passwd` must be run with `root` privileges. `ls -l` displays the letter `s` or `S` instead of the `x` for such programs at the user access bits—that is, in the first `rwx` group: a lowercase `s` if the execute bit is also set (the normal case), and an uppercase `S` if only the setuid bit but not the execute bit is set. The octal value of this bit for `chmod` is 4000:

```
user$ ls -l /bin/mount
-rwsr-xr-x    1 root   root    68508 Feb 25  01:11 /bin/mount

user$ ls -l /usr/bin/passwd
-rwsr-xr-x ... root root ... /usr/bin/passwd
```

If you want to find all programs that have the setuid bit set, you can use `find` with the `-perm -4000` option. In a default Ubuntu installation, the command finds around 15 setuid programs:

```
user find /usr/bin /usr/sbin/ /bin /sbin -perm -4000
/usr/bin/chsh
/usr/bin/chfn
/usr/bin/passwd
/usr/bin/gpasswd
/usr/bin/newgrp
/usr/bin/sudo
...
```

Attention: Security Risk!

The setuid bit can easily become a security risk—especially if other programs are started while the program is running. For this reason, you should avoid using the setuid bit whenever possible.

For script files (`bash`, Python, Tcl, and so on) the setuid bit is generally ignored on Linux! So it is impossible to give root privileges to a script by using a setuid bit.

Setgid bit for files

The *setgid bit* has a similar effect on programs as setuid. However, the group identification number of the file is used during the execution of the program, not the group ID (GID) of the current user. `ls -l` displays the letter `s` or `S` instead of the `x` for the group access bits—that is, in the second `rwx` group—for such programs. The octal value of this bit is 2000.

Setgid bit for directories

For directories, the setgid bit has a completely different meaning: if it is set, newly created files within this directory will be assigned the group of the directory—instead of the group of the person who creates the file, as is usually the case.

In reality, the setgid bit is used when multiple users are supposed to share a directory. In this case, it's useful to have new files be assigned to the common group and not to the currently active group of the user who creates the file. For this reason, the setgid bit in the previous section was used for the `/projects/xy` directory.

Sticky bit for directories

For directories in which everyone is allowed to change the files, the *sticky bit* makes sure that everyone is only allowed to delete their *own* files and not those of other users. For example, the bit is set for the `/tmp` directory. Every user is allowed to create temporary files in this directory. However, you should avoid allowing all users to rename or delete foreign files at will. `ls -l` displays the letter `t` instead of the `x` for all valid access bits in such programs. The octal value of this bit is 1000. The meaning of the sticky bit is specific to Linux! In other Unix variants, the bit may have a different meaning or no meaning at all:

```
user$ ls -ld /tmp/
drwxrwxrwt  ... root root ... /tmp/
```

Changing the special bits using chmod

The easiest way to set the special bits is to use `chmod` if you use octal code—for example, `chmod 2770 directory`. But of course, it's also possible to change the `owner+ bit` in the usual `chmod` syntax:

```
root# chmod u+s file   (sets the setuid bit)
root# chmod u-s file   (deletes the setuid bit)
root# chmod g+s file   (sets the setgid bit)
root# chmod g-s file   (deletes the setgid bit)
root# chmod +t  file   (sets the sticky bit)
root# chmod -t  file   (deletes the sticky bit)
```

In the past, you could use `chmod` only to set special bits in octal notation, but not to remove them again. Meanwhile this possibility exists, but under certain circumstances; for example, to delete the setgid bit from directories, you have to specify the octal code with five digits: `chmod 2770 dir` sets the setgid bit. `chmod 0770 dir` leaves the setgid bit unchanged! Only `chmod 00770 dir` or `chmod g-s dir` disables it again.

Representation of the special bits by ls

Interpreting the results of `ls -l` is no straightforward task:

- The setuid bit is indicated by the letters `s/S` instead of the `x` in the first `rwx` group.

- The setgid bit is indicated by the letters s/S instead of the x in the second rwx group.
- The sticky bit is indicated by the letters t/T instead of the x in the third rwx group.

The uppercase letters S and T are only applied if the corresponding execute bit is *not* set. This is usually an indication that the special bits are being used incorrectly.

File function or type

Within Linux, information about the function of a file is stored together with the access bits and the special bits. For example, it can be a normal file, a directory, a link, a block device, and so on.

Most file management programs hide this additional information.

However, there are a few programs that display this information as a numerical value. For an ordinary file, the complete specification is then 100000 (identifier for an ordinary file) plus N000 (special bits) plus xxx (access bits)—for example, 100760. The numeric codes for the file types are described in man 2 stat.

9.6.2 Owner and Group of New Files

This section deals with the question of what factors determine the access information of *new* files. To try this out easily, you can use the touch command. This command creates a new, empty file if the specified file doesn't yet exist.

Example

Say that the user michael creates a new file, myFile1. It should come as no surprise that this file again belongs to michael; after all, he just created it himself. The group membership was automatically set to michael. michael is the primary group of the user michael. Some distributions do not assign a separate group to each user but instead assign the users group to all users:

```
michael$ touch myFile1
michael$ ls -l myFile1
-rw-r--r--    1 michael   michael           0 Jun 14 16:45 myFile1
```

michael belongs to a number of other groups (groups command). To create a file that does not belong to the primary group, the active group must be changed first (newgrp command):

```
michael$ groups
michael adm admin cdrom docuteam dialout lpadmin plugdev sambashare
michael$ newgrp docuteam
```

```
michael$ touch myFile2
michael$ ls -l myFile2
-rw-r--r--    1 michael   docuteam          0 Jun 14 17:02 myFile2
```

Of course, `myFile2` could also have been created without a preceding `newgrp`. Then the group membership would have had to be changed afterward using `chgrp`. `newgrp` is useful when multiple new files are created that should automatically belong to a certain group.

Owner and group membership

The previous two examples show that new files are automatically owned by the user who creates them. Typically, the user's primary group is used as the group membership. However, there are two exceptions to this:

- If the user has made another of his groups the current group via `newgrp`, the new file belongs to that group.
- If the setgid bit is set in a directory (see the previous section), then files created in it are automatically assigned to the same group as the directory. The user's active group is not taken into account.

9.6.3 Access Bits of New Files (umask)

Access bits

Regarding the access bits, things are a bit more complicated. Linux actually provides that new files get the access bits `rw-rw-rw` (octal 666), so they may be read and modified by everyone. New directories and program files created by a compiler automatically get the access bits `rwxrwxrwx` (777), so they can be executed by anyone.

umask arithmetic

However, this basic setting would be too permissive for practical work with multiple users. That is why all Linux shells provide for the `umask` setting. This is a numerical value that specifies the bits that are subtracted from the standard access bits. You can determine the current setting of the `umask` value by using the command of the same name and, if necessary, change it for the course of the active session. Depending on the distribution, different default settings apply:

```
michael$  umask   (Arch Linux, Debian, Fedora, RHEL, SUSE)
0022
michael$  umask   (Ubuntu)
0002
```

Thus, many Linux distributions use the `umask` value 022 (`----w--w-`). For this reason, new files get the access bits 666 - 022 = 644 (`rw-r--r--`), while new directories and programs get the access bits 777 - 022 = 755 (`rwxr-xr-x`). The `umask` value 002, which is common in Ubuntu, is more liberal: new files get

the access bits 666 - 002 = 664 (rw-rw-r--), new directories 777 - 002 = 775 (rwxrwxr-x).

As each user typically uses their own group on Linux, the difference is negligible in practice. In both cases, custom files are readable by other users, but not modifiable.

umask for root

For root, the setting 0022 is valid for all distributions I tested:

```
root# umask
0022
```

Setting umask for your purposes

How must umask be set so that new files get the access rights rw-r----- = 640 and new directories rwxr-x--- = 750? The answer results from the subtraction of 777 minus the octal target value for directories—that is, 777 – 750 = 027:

```
michael$ umask 27
michael$ touch new-file
michael$ mkdir new-directory
michael$ ls -ld new*
-rw-r----- ... michael michael ... new-file
drwxr-x--- ... michael michael ... new-directory
```

umask configuration

The setting of the umask value is done in the configuration files of the shells. For bash, umask is usually set in /etc/profile or /etc/bashrc.

In some distributions, PAM (see Chapter 14, Section 14.6) takes care of setting umask. Among other things, the PAM module pam_umask analyzes the optional umask=xxx entry in the fifth column (*General Electric Comprehensive Operating System*; GECOS) of /etc/passwd. In addition, the settings from /etc/default/login and /etc/login.defs are taken into account. The /etc/login.defs file contains the umask setting on Ubuntu, for example (see the documentation on the UMASK and USERGROUPS_ENAB options contained in the file).

Individual users can make different settings in the .bashrc or .zshrc file. For example, if you want files you create to be read/executed only by group members and not by other users, you should use the following setting:

```
# in ~/.bashrc or in ~/.zshrc
umask 027
```

Who Is Allowed to Change Access Information?

Once a file is created, neither the owner nor the access bits change when it's edited by another user. Only the owner may change the group

membership and access bits. And only `root` is allowed to change the owner of a file. This means that it isn't possible for the owner of a file to give it to someone else, as it were.

9.7 Access Control Lists and Extended Attributes

Access control lists (ACLs)

The Unix-typical administration of users and groups as well as the access rights for directories and files based on them have proven themselves for decades. The concept is so simple that it can be understood after only a few hours. However, there are cases where this simple system is insufficient.

For this reason, a more finely meshed system for managing access rights was developed, based on so-called access control lists (ACLs). ACLs make it possible to set up any number of rules for each file or directory as to which users and groups are allowed to read or modify the file or directory and who is *not* allowed to do so—in deviation from the Unix access rights. ACLs thus act in addition to the standard access rights and can grant additional rights or revoke existing rights.

ACLs are active by default on popular Linux file systems. (In the past, ACLs had to be explicitly activated for some file system types by using the `mount` option, `acl`.)

The Samba file server is by far the most important program that really benefits from ACLs. It is able to mimic Windows access rights on Linux thanks to ACLs.

Limitations

Just because ACLs offer more options does not mean that they replace traditional rights management by any means! For experienced administrators of large networks, ACLs may provide additional security or at least simplify administration, but for most Linux users, ordinary rights management is absolutely sufficient. If you don't use the complex ACL system correctly, you may open security gaps. For this reason, there are currently hardly any distributions that use ACLs by default.

A key problem is that many Linux commands and programs do not process ACLs correctly. It can happen that a copied file suddenly lacks the ACL information of the original. Also, most file managers cannot properly display or modify ACLs. The Dolphin file manager in KDE is a positive exception.

Extended attributes (EAs)

Closely related to ACLs are extended attributes (EAs), which allow you to store additional attribute-value pairs for each file. Thus, for example, you can assign the `charset` attribute with the `utf8` setting to a text file in order to store the character set used. Of course, this only brings advantages if

there are also programs that analyze this information. Depending on which file system you use, EAs must also be enabled by a corresponding `mount` option—for example, via `user_xattr` for the `ext` file system.

For more background and details on using ACLs and EAs, see the `man` pages on `acl`, `getfacl`, `setfacl`, `attr(5)`, `getfattr`, and `setfattr`.

Prerequisites for this section

In the following examples, I assume that the `attr` package with the `attr`, `getfattr`, and `setfattr` commands is installed and that you are working with a file system that has ACLs and EAs enabled. This is the case by default for the `ext4`, `btrfs`, and `xfs` file systems. If ACL is not available, you will receive errors like `Operation is not supported` when you execute `setfacl`. The solution to this is usually the `acl` option in the `mount` command or in the `/etc/fstab` file.

9.7.1 Access Control Lists

getfacl

Even on a file system with ACLs, the default access rights, often referred to as the *minimum ACL*, usually apply. `getfacl` displays these rights in ACL terms:

```
user$ touch file1
user$ getfacl file1
# file: file1
# owner: kofler
# group: kofler
user::rw-
group::r--
other::r--
user$ ls -l file1
-rw-r--r-- 1 kofler kofler ...  file2
```

setfacl

Using `setfacl`, you can now define additional access rules. The following commands grant the user `gladys` and all members of the `docuteam` group read and write access to the file, but deny the user `catherine` any access:

```
user$ setfacl -m gladys:rw file1
user$ setfacl -m g:docuteam:rw file1
user$ setfacl -m catherine:- file1
```

The rights list of `getfacl` is now a bit longer. `ls` shows the ACL mask for group member access rights. The access letters are followed by the `+` character to indicate that there are ACL rules:

```
user$ getfacl file1
# file: file1
```

```
# owner: kofler
# group: kofler
user::rw-
user:gladys:rw-
user:catherine:---
group::r--
group:docuteam:rw-
mask::rw-
other::r--
user$ ls -l file1
-rw-rw-r--+ 1 kofler kofler ... file1
```

A typical use of ACLs is that you want to give a certain user access to your files, but without making the files immediately available to all other users (of a certain group).

Typically, you would now have to ask the administrator to create a new group that includes you and the other users with whom you want to share the files. With ACL, you simply execute `setfacl -m user:rw file`.

ACL mask

The ACL mask limits the rights provided by ACL rules. For example, if you set the ACL mask to `r`, no ACL rule can give a user write or execute permissions. Thus, the ACL mask takes precedence over the ACL rules. However, it has no effect on the rights that result from the traditional access rights for the owner of the file or for group members of the file. Whenever an ACL rule is changed by `setfacl`, the mask is automatically recalculated so that all other ACL rules can be complied with. This mask is displayed by `getfacl` and is also taken into account by `ls -l`.

You can explicitly set the mask by `setfacl -m m:RWX file` to limit ACL permissions. Note that you need to replace `RWX` with the desired access bits. However, your own mask is only valid until you define a new ACL rule. This will automatically recalculate the ACL mask (unless you prevent this by using the `-n` option).

Standard ACL

For directories, you can specify a second set of rules for the default ACL. The default ACL does not control access to the directory but is considered a pattern for new files. Each file that is newly created within the directory inherits the directory's default ACL, so to speak. For many ACL applications, a new directory with a cleverly chosen default ACL serves as a starting point.

ACL compatibility

The biggest obstacle to the wider adoption of ACLs is that many standard commands and almost all application programs simply ignore ACLs. If you simply copy a file with ACL rules using `cp`, the copy will lose all ACL rules. The same applies if you open the file in an editor, such as LibreOffice or

GIMP, and save it under a different name. For cp, the -p option solves this problem, but most other commands and programs lack comparable options or ACL-compliant behavior.

Backups are another issue as tar and rsync eliminate ACL rules. The file systems of CDs and DVDs does not provide for ACLs, so this information is lost there as well. There are two ways out: you can either use the ACL-compatible star variant instead of tar or create an additional text file containing the ACL rules of all files before the backup. After the backup, you can then restore the ACL rules using that file:

```
user$ getfacl -R --skip-base . > acl-backup     (backup ACL rules)
user$ setfacl --restore=acl-backup              (restore ACL rules)
```

9.7.2 Extended Attributes

setfattr and getfattr

The following examples show how you can use setfattr to store attributes and how to read them using getfattr:

```
user$ touch file2
user$ setfattr -n user.language -v en file2
user$ setfattr --name=user.charset --value=utf8 file2
user$ getfattr -d file2
# file: file2
user.charset="utf8"
user.language="en"
```

getfattr usually returns only attributes whose names start with *user*. If you want to see other attributes, you must specify their names using -n or their patterns by using -m:

```
user$ getfattr -n security.selinux -d tst
# file: tst
security.selinux="user_u:object_r:user_home_t:s0^000"
```

EA compatibility

Unfortunately, there are hardly any programs at the moment that preserve extended attributes during copy, archiving, or similar processes. Even cp -p ignores the attributes. For backups, the best approach is similar to ACLs in that you create a file with all EAs prior to the backup. Then you can use this file to restore the EAs later:

```
user$ getfattr -R . > ea-backup          (backup attributes)
user$ setfattr --restore=ea-backup       (restore attributes)
```

9.7.3 Capabilities

One of the most interesting uses of extended attributes is the ability to specify, for executable files, which operations are allowed for the program—that is, which *capabilities* the program has. This would allow fewer programs to be marked with the setuid bit and in this way increase the security of Linux distributions.

Prerequisites

For capabilities to work, two conditions must be met:

- The `libcap` library must be installed (`/lib/libcap*` or `/lib64/libcap*`).
- The file system must support EAs because the capability data is stored as EAs. This is the case by default for the most common Linux file systems.

getcap and setcap

To administrate capabilities, you need the `getcap` and `setcap` commands. They are mostly included in the `libcap-ng-utils` or `libcap-progs` package. You need to install this package separately for many distributions.

More and more distributions are using capabilities to grant commands the right to send network packets or generate new UIDs/GIDs without `root` rights:

```
root# getcap /usr/bin/*     (results for Fedora 38)
/usr/bin/arping     cap_net_raw=p
/usr/bin/clockdiff cap_net_raw=p
/usr/bin/newgidmap cap_setgid=ep
/usr/bin/newuidmap cap_setuid=ep
```

In the past, the insecure SetUID bit was set for such commands—or the application was restricted to `root`.

9.8 The Linux Directory Structure

Filesystem Hierarchy Standard

A typical Unix system consists of thousands of files. During the development of Unix, certain rules have emerged about in which directories what files are usually stored. These rules were adapted to the specifics of Linux and summarized in a separate document: the *Filesystem Hierarchy Standard* (FHS; see *https://refspecs.linuxfoundation.org/fhs.shtml*). Most Linux distributions adhere to this standard.

The information summarized in this section will give you some initial guidance. I have taken not only the FHS into account, but also the common habits of popular Linux distributions.

The file system starts with the root directory. There are usually no files there, only directories:

- **/bin**
  Contains elementary Linux commands for system administration that can be executed by all users. Other programs are located in /usr/bin. In modern distributions, /bin is simply a link to /usr/bin; the separation between /bin and /usr/bin has thus been removed.
- **/boot**
  Contains files that are used to boot the system (usually via GRUB). In most distributions, the kernel is also located here.
- **/dev**
  Contains all device files. Almost all hardware components—such as the serial interface or a hard disk partition—are accessed via so-called device files. These are set up dynamically by the udev system (see Section 9.9). In most distributions, the /dev directory is located in a RAM disk; that is, the contents of the directory are not preserved when the computer is rebooted.
- **/etc**
  Contains configuration files for the entire system. Within /etc, there are numerous subdirectories that organize the configuration files into groups—for example, /etc/apt for files of the apt package management system. A large number of files from /etc are described in the configuration chapters of this book. You can also take a look at the index (letter E)!
- **/home**
  Contains the home directories of all regular Linux users. The home directory is the directory in which the user is automatically located after logging in and to whose files he or she has unrestricted access rights. As so often, root is a special case: its home directory is /root.
- **/lib[64]**
  Contains some shared libraries or symbolic links to them. The files are needed to run programs. /lib/modules contains kernel modules that are dynamically enabled or disabled during operation. Other libraries are located in /usr/lib[64]. The /lib/firmware directory contains the firmware of various hardware components (e.g., a Wi-Fi controller).

  In current distributions, /lib is a link to /usr/lib. This link places all libraries centrally in the /usr directory.
- **/lost+found**
  Exists only in ext file systems. The directory is usually empty. If it does contain files, then these are file fragments that could no longer be allocated when fsck attempted to repair the file system. In other words,

sectors were found, but it is unclear to which file each sector once belonged. Instead of simply deleting such file fragments, `fsck` copies them to the `lost+found` directory.

`fsck` is automatically executed during system startup if Linux was not shut down properly (power failure, crash, etc.) or if the file system has not been checked for a long time. The goal of `fsck` is to restore the file system to a clearly defined state.

- **`/media`**
  Contains subdirectories like `cdrom` or `<usb-drive-name>` where external file systems are mounted. Traditionally, `/mnt` was used for this, but in recent years `/media` and finally the `/run/media/<username>/<data_medium name>` directory have become established instead.
- **`/opt`**
  Is intended for add-on packages, but is rarely used by the common distributions—probably because it is unclear how add-on packages differ from normal packages.
- **`/proc`**
  Contains subdirectories for all running processes. These are not real files! The `/proc` directory merely reflects the administration of the processes within Linux.
- **`/root`**
  Contains the files of the `root` user—that is, the system administrator.
- **`/run`**
  Contains files with the process IDs and other information of some system services for many current distributions. In the past, these files were stored in the `/var/run` directory.

  The `/run/lock/` subdirectory contains locking files. In older distributions, you will find the locking files in `/var/lock` instead.

  Many distributions store either the entire `/run` directory or at least individual `/run` subdirectories in a RAM disk. The predominantly very small files in `/run` are therefore never physically stored on a hard disk or SSD and are lost when the computer is rebooted.
- **`/sbin`**
  Contains commands for system management. A common feature of all programs stored in it is that they may only be executed by `root`. In modern distributions, `/sbin` is a link to `/usr/sbin`; all system administration commands are now stored in `/usr/sbin`.
- **`/share`**
  Sometimes contains architecture-independent files—that is, files that

are independent of the processor. The correct location is actually /usr/share.

- **/srv**
  Contains data for server processes in some distributions (Fedora, RHEL)—for example, in /srv/www files of the web server or in /srv/ftp files of the FTP server.
- **/sys**
  Contains the sysfs file system. Like the proc file system, it provides information about the state of the computer.
- **/tmp**
  Contains temporary files. However, temporary files are often stored in /var/tmp.
- **/usr**
  Contains all application programs, the complete X system, the source code for Linux, and so on. The contents of this directory usually change only during package installations and updates. The /var directory is intended for mutable files, but Table 9.9 gives a brief description of the most important subdirectories of /usr.
- **/var**
  Contains mutable files. Important subdirectories include docker (docker files), lock (locking files to protect access to devices), log (logging files), mail (email files, often also in /var/spool/mail), mysql (MySQL database files), run (files with process IDs of some system services), and spool (cached print files).

So the basic structure of the root-level directories is quite easy to understand. The problems only start with the subdivision of /usr and /var into countless subdirectories. Basically, many directories are named in the same way as in the root level—for example, bin for executable programs.

The problem here is that there are several groups of executable programs: text-based commands, X programs, and so on. In the past, there were individual directories for these groups of programs, such as /usr/bin/X11 for programs with graphical user interfaces. By now, most distributions try to install all programs into *one* directory—namely, /usr/bin. Symbolic links establish compatibility with past standards.

Directory	Contents
/usr/bin	Executable programs
/usr/games	Games; possibly links to /usr/share/games

Table 9.9 /usr Directories

Directory	Contents
/usr/include	C include files
/usr/lib[64]	Various libraries, also countless subdirectories for C compilers, various other programming languages, large program packages such as emacs or LaTeX, and so on
/usr/local	Applications and files that do not directly belong to the Linux distribution or were installed later
/usr/sbin	Programs that can only be executed by root
/usr/share	Architecture-independent data (e.g., Emacs Lisp files, Ghostscript character sets, etc.), documentation (/usr/share/doc)
/usr/src	Source code for Linux and possibly other programs

Table 9.9 /usr Directories (Cont.)

9.9 Device Files

Not only files and directories are managed in the Linux file system, but also so-called devices. These are specially marked files that do not store data, but rather connect to the Linux kernel (see Table 9.10).

Major and minor device number

Devices enable access to many hardware components of the computer, such as hard disks/SSDs, serial and parallel interfaces, the main memory (RAM), and so on. Devices are characterized by three pieces of information: the major device number, the minor device number, and the type of access (block-based or character-based).

The major device number specifies which Linux kernel driver is responsible for management. Most drivers are listed with their major device number on the following page: *https://www.kernel.org/doc/Documentation/admin-guide/devices.txt*.

In many drivers, the minor device number is used to differentiate between different (related) individual devices—for example, in the driver for hard disks between different partitions.

The access type indicates whether the devices are buffered (this is the case for all block-based devices such as hard disks and the like) or not (this applies to character-based devices such as the serial interface).

If you use `ls -l` to view the table of contents of `/dev`, the device numbers (major and minor) are output instead of the file size. The first character of the access bits is `b` or `c` (block-based or character-based):

```
user$ ls -l /dev/sda?
brw-rw---- 1 root root 8, 1 ... /dev/sda1
brw-rw---- 1 root root 8, 2 ... /dev/sda2
...
```

Internal details

Inside Linux, there are only so-called inodes in the `/dev` directory: these are the smallest administrative units of a file system, but not real files with content. New device files can be set up using the `mknod` command. In practice, however, this is rarely necessary because the `udev` system takes care of it automatically. The major and minor device numbers are combined to form a 64-bit number.

For security reasons, only `root` or members of a specific group are allowed to access many devices. To allow other users to access these devices, you need to add the user to this group.

Some device files have a special function: thus, `/dev/null` serves as a "black hole" into which data can be sent that will disappear there forever—for example, to redirect command output that should not be displayed. `/dev/zero` is an inexhaustible source of 0-bytes, sometimes used to fill files up to a given size with zeros. `/dev/random` and `/dev/urandom` supply random numbers.

udev system

In the past, distributions generated thousands of device files during installation. At most, a few hundred files are actually used; however, there are different device files on each computer—depending on the hardware equipment. The `udev` system provides a workaround. The background program `udevd`, or `systemd-udevd` in current distributions, detects all hardware components connected to the computer and creates the necessary device files as required. `udevd` or `systemd-udev` is started by the init process. The configuration is done by the files in the */etc/selinux* directory.

As part of the effort to get Linux up and running faster, the `devtmpfs` file system has been added to `udev`. This temporary file system maps the `/dev` directory. During the boot process, `devtmpfs` can be used without the overhead of the full `udev` system.

Device	Meaning
`/dev/console`	The currently active virtual terminal
`/dev/disk/*`	Additional links to hard disk and partition devices
`/dev/dri/*`	Direct rendering infrastructure (3D graphics with X)
`/dev/dsp*`	Access to the sound card (digital sampling device)

Table 9.10 Important Device Files

Device	Meaning
/dev/fb*	Frame buffer (graphics card)
/dev/input/*	Keyboard and mouse
/dev/kmem	Memory (RAM) in core format (for debuggers)
/dev/mapper	Mapping files for LVM, crypto containers, and so on
/dev/md*	Meta devices (RAID, etc.)
/dev/mem	Memory (RAM)
/dev/mixer*	Access to the sound card
/dev/nvme*	SSDs that support the NVME protocol
/dev/port	IO ports
/dev/pts/*	Virtual terminals according to Unix 98
/dev/ptyp*	Virtual terminals under X (master)
/dev/ram	RAM disk
/dev/raw1394	Direct access to Firewire devices
/dev/scd*	CD/DVD drives
/dev/sd*	SATA/SCSI/USB/Firewire media
/dev/shm	POSIX shared memory
/dev/snd	ALSA sound (link to /proc/asound/dev)
/dev/sr*	SCSI/SATA/USB/Firewire CD/DVD drives
/dev/tty*	Virtual terminals in text mode
/dev/ttyp*	Virtual terminals under X (slave)
/dev/ttyS*	Serial interfaces (modem, mouse, etc.)
/dev/usb/*	USB devices (see also /proc/bus/usb)

Table 9.10 Important Device Files (Cont.)

Chapter 10
Process Management

This chapter describes the way Linux handles processes. In the course of this chapter, you will learn

- Which options you have to start programs and to terminate them again (if necessary, also by force)
- How to run a program with root privileges
- What demons are
- How to start programs automatically at specific times using cron or systemd timers

10.1 Starting, Managing, and Stopping Processes

Programs, commands, processes, tasks

This chapter mainly deals with processes. At the operating system level, a *process* is responsible for the execution of a program or command. This sounds like a rather trivial task; however, as many programs and background services run in parallel, it isn't at all easy to distribute the computing time between all programs fairly or in such a way that it makes sense.

> **Programs and Commands**
>
> As a matter of fact, a program or command is just an executable file. Thus, a program file differs from other files in that the execute bit `x` is set. Within Linux, there is no distinction between a program like Firefox or a command like `ls`. In everyday language, however, text-based programs such as ls are often referred to as *commands*.

The program file doesn't become an active process managed by the Linux kernel until it is started. Seen in this light, the heading of this section should actually read "Starting Programs and Commands, and Managing and Stopping Processes".

***.exe files**

Every now and then, the question arises where the `*.exe` files are located on Linux. The usual answer is: there are no `*.exe` files. Executable programs are identified by the access bit `x`; the file identifier `*.exe` known from Windows is thus superfluous.

This answer is not quite correct insofar as there can actually be isolated `*.exe` files on Linux. These are programs that were developed in the C#

programming language and use the Mono library for execution. The Mono library is in turn an open-source implementation of Microsoft's .NET framework.

10.1.1 Launching Programs

Launching in a desktop environment

In a desktop system, you usually launch programs from a menu or by clicking an icon. The Alt+F2 or Windows keyboard shortcuts are an additional way to quickly launch programs.

Text console, terminal window

Alternatively, you can start programs in a terminal window or in a text console. To do this, you simply need to type the name of the program and press Enter. Especially Linux professionals often choose this method because it is faster to type a few letters than to search for the program in branched menus. & causes the program to run in the background and not block the terminal.

Usually it's enough to simply specify the name of the program. The shell interpreter then searches for the program in all directories specified in the PATH environment variable. The following lines show a typical setting of this variable:

```
user$ echo $PATH
/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin
```

If you want to start a program that isn't located in any of these directories, you must specify the full path. This also applies to programs in the current directory! Here the path is simply specified by a dot, such as `./myprogram`.

xdg-open

Provided a desktop system such as GNOME or KDE is installed, you can also open images and documents via `xdg-open filename`. `xdg-open` automatically starts the desktop program associated with the file type. In some distributions, you can simply use `open` instead of `xdg-open` (as on macOS).

10.1.2 Foreground and Background Processes

When you launch programs in the start menu of your desktop, they run as *background processes*—that is, without interfering with each other. You can launch more programs without waiting for the previously started programs to finish.

The behavior is completely different when you run a program in a text console or terminal because then the program is launched as a *foreground process*. Before you can enter the next command in the terminal, you must wait for the program that was launched last to finish.

But you can also start programs in the background in text consoles or terminal windows. To do this, you simply want to specify the & character at the end of the command:

```
user$ firefox &
```

From front to back

If you forget to specify the & character, you can also convert the program to a background process afterward. What you need to do is interrupt the program execution using Ctrl+Z and continue the program by using bg:

```
user$ firefox
<Ctrl>+<Z>
+  Stopped  firefox
user$ bg
+ firefox &
```

If you use the fg command instead of bg, the program will continue running as a foreground process.

For some commands, various text outputs interfere with the background execution. However, you can easily suppress them by redirecting them to /dev/null. For example, the following command sets up a file system in the background:

```
root# mkfs.ext4 /dev/sdc1 > /dev/null &
```

10.1.3 List of All Running Processes (ps and top)

ps

You can generate a list of currently running processes very easily via ps. If you enclose the process name in square brackets, then it's a process of the kernel. Without any additional options, ps shows only your own processes that you have started in the terminal. ps can be controlled by countless options, many of which are specified without the otherwise usual preceding minus sign. In the following example, the list of processes has been greatly shortened for reasons of space. A typical Linux system with a graphical user interface usually has well over 100 processes running at the same time:

```
user$ ps ax
  PID TTY      STAT   TIME COMMAND
    1 ?        Ss     0:00 init [2]
    2 ?        S      0:00 [kthreadd]
    3 ?        S      0:00 [ksoftirqd/0]
 ...
 3064 pts/2    S      0:39 emacs command.tex
 3151 pts/2    S+     1:23 /bin/sh ./lvauto
```

```
 3735 pts/4    S      0:00 su -l
 3740 pts/4    S+     0:00 -bash
```

top Often `top` is more useful than `ps`. This command orders the processes according to how much load they put on the CPU and displays the currently active processes first. The program also provides an overview of the current memory requirements, and so on. The process list is updated every few seconds until the program is terminated using Q. The following lines show a web server that is almost idle:

```
top - 20:50:38 up 11 days, 12:18, 1 user, load average: 0.10, 0.09, 0.08
Tasks: 114 total,   1 running, 113 sleeping,   0 stopped,   0 zombie
Cpu(s):  2.6%us,  0.2%sy,  0.0%ni, 96.8%id,  0.4%wa,  0.0%hi,  0.0%si,
0.0%st
Mem:   4049808k total,  1580060k used,  2469748k free,   203396k buffers
Swap:   521212k total,        0k used,   521212k free,   804400k cached

  PID USER      PR  NI  VIRT  RES  SHR S %CPU %MEM    TIME+  COMMAND
32601 www-data  20   0  346m  32m 4296 S    5  0.8   0:00.15 apache2
32592 www-data  20   0  344m  31m 4324 S    3  0.8   0:00.47 apache2
  851 mysql     20   0 1403m  53m 7948 S    0  1.4   6:40.10 mysqld
    1 root      20   0 24336 2184 1272 S    0  0.1   0:01.07 init
  ...
```

The value in the PID column indicates the process number. If you know this number, you can forcibly stop out-of-control programs or background processes using the `kill` command.

Processes can assume different states. The two most common states are R (*running*) and S (*sleeping*; i.e., the program has nothing to do at the moment and is waiting for input). Programs can also be temporarily interrupted and then have the state T (*stopped*). `top` also accepts commands interactively. This means you can stop processes (K; `kill`) or change their priority (R; `renice`).

htop and iotop A convenient alternative to `top` is the `htop` command, which must be installed separately in most distributions. Among other things, it allows horizontal and vertical scrolling in the process list.

If you want to track not the CPU and memory usage but the accesses to hard disks and other volumes, you should start the `iotop` command instead of `top`. The `-o` option enables you to restrict the output to processes that actually cause IO activity. `-u` restricts the output to your own processes. `iotop` is part of the package of the same name, which usually has to be installed separately.

Determining the process number

If you want to know the program name and determine the associated process number (PID), `pidof` can help you. If there are several processes with the same name, `pidof` returns a whole list of numbers:

```
root# pidof sshd
4429 4393 4172
```

Sometimes it is also useful to determine which programs access a particular file or directory. You can determine the corresponding process numbers by using `fuser`. A directory is considered used even if a program has been started in it. The following command shows that the `zsh` shell and the Docker program use the current directory:

```
user$ fuser -v .
                     USER        PID ACCESS COMMAND
                     kofler     4211 ..c.. zsh
                     kofler     7515 ..c.. docker
```

Note that a file access can be detected only if the program has actively opened the file. This is usually not the case with a text editor: the editor opens the file for loading, but then closes it again. It only opens the file again for a short time before saving.

PID files

Some background processes store a PID file in the `/var/run` directory—for example, `/var/run/httpd.pid`. This file contains the process number in the first line; further lines may contain additional information, such as the network interface. PID files enable the targeted termination of a specific process by the init system, even if multiple processes with the same name exist.

Graphic variants to top

There are also graphical alternatives to the `top` and `htop` commands, such as `ksysguard` (KDE) or `gnome-system-monitor` (GNOME). Conky provides incomparably more display and configuration options. On the internet, you can find Conky configurations that are true works of art and turn a bland desktop background into an original and useful information center. Unfortunately, setting up a Conky configuration involves a lot of work. The configuration files offered on the internet are mostly outdated or fit only for a special distribution.

Server monitoring

If you don't want to monitor your own computer but several external servers, the best option is to use tools such as Nagios or Munin. However, their configuration is complex and requires experience in server administration. A simpler alternative for a single server is the Linux Dash project. After an uncomplicated installation, a page that can be opened in the web browser summarizes the most important status information.

10.1.4 Process Hierarchy

Internally, the PID number of the parent process is also stored with each process. This information enables the representation of a process tree, at the top of which there is always the `systemd` program. This program is started immediately after the kernel has been loaded (see Chapter 20). The `pstree` command, which sometimes has to be installed separately, shows the hierarchy of the process tree. The `-h` option can be used to highlight the parent processes of the currently running process in bold.



Figure 10.1 Process Overview with pstree

10.1.5 Terminating Processes by Force (kill and xkill)

Usually, a process runs until the regular end of the program. But unfortunately, it also happens on Linux that programs contain errors, can no longer be stopped, and may possibly consume more and more memory and CPU capacity. In such cases, the process must be terminated by force. For text-based commands, Ctrl+C helps in most cases. This will terminate the program immediately.

kill

The `kill` command sends signals to a running process specified by the PID number. You can determine this number via `top` or `ps`. To end a program "politely," you need to use signal 15. `kill` uses this signal by default. If this doesn't help, signal 9 must be used (here for process 2725):

```
user$ kill -9 2725
```

You can use `kill` only for your own processes. Only `root` may also terminate foreign processes.

top

You can also use `top` to terminate processes: simply enter K and then the process number and the desired signal!

killall

`killall` is more convenient in that not a process number but the program name can be specified. However, *all* processes of this name will then be terminated:

```
root# killall -9 firefox
```

xkill

Provided your graphics system uses the X Window System and not the modern Wayland library, there is an even more convenient way to terminate a program: for this purpose, you need to run `xkill` in a terminal and then click the window of the program you want to stop. Again, signal 9 is then sent to the process. On KDE, you can also start `xkill` using Ctrl+Alt+Esc. If this happens by mistake, you can cancel `xkill` with Esc.

Stubborn cases

Sometimes `xkill` closes the window, but the process or parts of it continue to run. Use `top` or `ps` to make sure that the program is really finished. If necessary, you can also use `kill -9 <pid>`.

Blocked keyboard or mouse

It becomes really unpleasant when a desktop program not only hangs, but also takes over the keyboard and mouse focus or blocks the graphics system in some other way. The computer then no longer responds to any input. In such cases, the magic key combination Ctrl+Alt+F# sometimes can help, which switches to the # text console. There you can log in, search for, and quit the relevant program by using `top`.

If the keyboard is completely blocked, it's still possible to log in through a network via `ssh` and run `kill` that way. Of course, this variant is only possible if you work in a local network and an SSH server is running on your computer.

Limiting the process size

For programs started from a shell (such as all commands executed in a terminal), you can use the `ulimit` command to limit the maximum memory consumption, the maximum size of created files, and so on. `ulimit` is usually set in `/etc/profile`.

10.1.6 Distribution of Compute Time (nice, renice, and ionice)

In everyday Linux operation, the compute capacity is usually sufficient to execute all running processes without delays. However, when Linux is busy with computing-intensive processes—for example, while compiling a large program—it tries to distribute the available computing time evenly to all processes.

In some cases, it makes sense to deliberately allocate more or less compute time to a process. The `nice` command, which can be used to launch programs with reduced or increased priority, is used for this purpose. Here, the desired priority is passed to `nice`, ranging from 19 (very low) to -20 (very high). By default, processes are started with priority `0`. In the following example, a backup program is launched with lower priority so that it doesn't interfere with other processes. It doesn't matter if the backup takes a few seconds longer:

```
user$ nice -n 10 ./my-backup-script
```

renice

`renice` can also be used to change the priority of processes that are already running. The process ID that was previously determined using `top` or `ps` must be specified as a parameter. Details about `renice` can be found on the `man` page. `top` is also able to interactively change the priority of a process; to do this, just press R.

However, only `root` can start programs with a higher priority than `0` or increase the priority of an already running process.

ionice

Often it isn't the CPU but the data medium that is the limiting factor in the execution of programs. For example, if you want to prevent a backup script from taking up the entire I/O capacity of the computer and thus slowing down other, perhaps more time-critical processes, you can run it with reduced I/O priority using `ionice`. The following command reads a logical volume, compresses its contents, and saves it to an image file:

```
root# ionice -c 3 cat /dev/vg1/snap | lzop -c > /backup/image.lzo
```

10.1.7 Input and Output Redirection and Pipes

Almost all text-based commands expect input via the standard input channel (the keyboard by default) and send output to the standard output channel; the resulting text is then simply displayed in a terminal. Both input and output can be redirected, providing many options. For example, the following command saves the list of all files in the `xy` directory in the `z` file:

```
user$ ls xy > z
```

Pipes allow the output of a command to be used as input for the next command. In the following example, `egrep` filters out from the list of all installed packages those that contain the "mysql" or "mariadb" strings in uppercase or lowercase. `sort` finally sorts this list:

```
user$ dpkg -l | egrep -i 'mysql|mariadb' | sort
ii  libmysqlclient21:amd64  8.0.23   amd64 MySQL database client library
ii  mysql-client            8.0.23     all MySQL database client ...
...
rc  mariadb-common          1:10.3.25  all MariaDB common metapackage
```

In other words, the output of the `dpkg` command is passed to `grep` thanks to the first | character, and its output is passed to `sort` via the third | character. (The second | character is interpreted by `egrep` in the sense of *or*). For more details and examples on input and output redirection, see Chapter 7, Section 7.4.

10.2 Running Processes under a Different Identity (su)

There are two restrictions on the execution of programs by ordinary users:

- Ordinary users are allowed to run only those processes where the access rights (owner, group, r, and x access bits) allow it. For ordinary programs, this is not a limitation. However, there are some system administration commands in the `/usr/sbin` directory, for example, that can only be started by `root`.
- Processes belong to the user who started them, as it were. This means that a process is allowed to access the same files as the user. Put another way, files that you as a user are not allowed to modify must also not be modified by programs that you start. Files newly created by the process also belong to the user who started the program (see also Chapter 9, Section 9.7.2).

For these reasons, you cannot perform many administrative tasks as an ordinary user. The easiest way out is to log in as `root`. However, I have

already pointed out several times in this book that it is not a good idea to work as root all the time. The risk is simply too great that you could cause damage by mistake. For this reason, some distributions block the root login completely. Thus, a direct root login is impossible on Ubuntu.

This section describes how you can still perform administrative activities using su without logging out as an ordinary user. Section 10.3 describes an alternative approach using sudo, which has many advantages over su. Finally, Section 10.4 presents another way using the PolicyKit program, which has established itself primarily on desktop systems.

10.2.1 The su Command

In many cases, you just need to quickly execute a command as root. It would be inconvenient to leave the currently active desktop system or terminal window and log in again as root.

The easiest way to change the user within a terminal window is to use the su name command. If you don't execute the command as root, you'll be asked for the password of the respective user. Inside the terminal, you can then execute commands under the changed name until you return to normal mode via exit or Ctrl+D.

The following lines show how an ordinary user logs in briefly as root, mounts a disk partition in the directory tree as root, and then logs out again as root and continues working normally:

```
user$ su -l root
Password: *********
root# mount -t ext2 /test /dev/sda7
root# <Ctrl>+<D>
logout
user$ ls /test
```

For su to be a full replacement for a root login, you must use the -l option! This ensures that all login start files are imported, which is necessary for the correct definition of PATH, among other things.

pkexec

Depending on the configuration and graphics system (like Wayland), desktop programs cannot be run in a su session. Rather, most distributions use the pkexec command to launch programs with admin privileges in the desktop environment. Behind the scenes, pkexec again uses PolicyKit (see Section 10.4).

Setuid and setgid access bits

The setuid and setgid access bits provide another way to mark certain programs so that anyone can run them as if he or she were root or another user

or member of another group. However, this removes an important protective mechanism: the setuid bit applies to *all* users, regardless of whether they know the `root` password (for `su`) or belong to a `sudo` group. For more information about the setuid and setgid access bits, see Chapter 9, Section 9.6.1.

10.3 Running Processes under a Different Identity (sudo)

`sudo` takes a different approach than `su`. The program, after appropriate configuration, allows certain users to run programs with `root` privileges. For security reasons, the user's *own* password must be entered again—that is, *not* the `root` password. This is especially useful on servers because it eliminates the tedious and insecure management of a common `root` password for multiple administrators.

`sudo` then runs these programs as if they were started by a different user (default: `root`). This allows individual users to perform admin tasks or run system-critical commands without having to know the `root` password. `sudo` logs all executed commands and failed authentication attempts in the syslog.

Depending on the configuration, the `sudo` authentication remains valid for a few minutes. If you execute another command using `sudo` within this period, you will not be asked for the password again. The timeout can be changed in `/etc/sudoers` using the `timestamp_timeout` keyword.

Configuration

The configuration of `sudo` is done through the `/etc/sudoers` file. In simple terms, the file describes in three columns which users are allowed to run which programs from which computer. The following line means that user `catherine` is allowed to execute the `/usr/sbin/parted` command. Because of the first `ALL`, this rule applies regardless of the hostname. `=(ALL)` means that `catherine` may execute the command under any account—as `root`, but also as, for example, `peter` or `ann`:

```
# in /etc/sudoers
catherine ALL=(ALL) /usr/sbin/parted
```

If the first column of `sudoers` is prefixed with the `%` character, the entry applies to all members of the specified group. Instead of `(ALL)`, the setting `(All:ALL)` is often used as well. This means that commands can be executed in any account and in any group. Other syntax variants are described in `man sudoers`.

Modify /etc/sudoers Using visudo!

For security reasons, /etc/sudoers should only be edited using the visudo command (see also its man page)! visudo performs a syntax test before saving and thus ensures that you do not exclude yourself from further admin work due to a faulty sudoers file. This is especially important for distributions like Ubuntu, which do not provide a root login.

visudo, as the name suggests, typically uses the vi editor; if you prefer a different editor, you must specify its exact path in the EDITOR environment variable before running visudo—export EDITOR=/usr/bin/jmacs, for example.

The configuration of /etc/sudoers provides a lot more syntactic options than have been mentioned here. You should therefore read the man pages on sudo and on sudoers! You can find even more details on the sudo homepage at *https://sudo.ws*.

Usage

Catherine can now run parted as follows, authenticating herself with her own password:

```
catherine$ sudo /usr/sbin/parted /dev/sda
Password: xxxxxx
```

For parted, the full path must be specified if parted is not in one of catherine's PATH directories. parted is automatically run in the root account. You can select a different account by using sudo -u <account>.

sudo without a password

It's possible to allow a specific user to run sudo without specifying a password. To do this, you need to add a line in sudoers according to the following pattern:

```
maria ALL=(ALL) NOPASSWD: ALL
```

This is a security risk, of course, but if you often need to perform admin tasks, you will appreciate the convenience gained. Note that the NOPASSWD tag is only valid if there are no other sudoers lines that require a password from the same user. This also applies to group entries, such as %admin.

In real life, it's often useful to exempt only the invocation of individual commands from authentication. The following setting allows peter to execute the sudo docker command without specifying a password:

```
peter  ALL=(ALL:ALL) NOPASSWD: /usr/bin/docker
```

Trouble with input/output redirection

If you use input and output redirection in a command executed using sudo, it's executed directly by the shell, not by sudo—and thus only with your local privileges, not with the root privileges. This causes errors:

```
user$ sudo ls /etc > /etc/ls-out.txt
-bash: /etc/ls-out.txt: No authorization
```

You can work around this problem by passing a shell to sudo and executing the desired command in it:

```
user$ sudo sh -c 'ls /etc > /etc/ls-out.txt'
```

Running shell scripts using sudo

Sometimes it happens that you want to execute multiple commands automatically with a script via sudo. The correct syntax then looks as follows:

```
user$ sudo bash script.sh
user$ sudo ./script.sh      (if the script starts with shebang)
```

Executing commands using sudo

For more extensive admin tasks, it becomes increasingly annoying to prefix sudo to every command. It's more elegant to switch to the root mode by using sudo -s. All other commands are executed as if by root. You can exit this mode using exit or Ctrl+D.

```
user$ sudo -s     (switch to root mode)
root# cmd1
root# cmd2
root# cmd3
user$ exit        (leave root mode)
```

Graphic programs

In the Xorg graphics system, you can easily run graphical programs using sudo. When Wayland is active, different rules apply depending on the version. In current distributions, sudo works like under X without restrictions. In older distributions, on the other hand, you must also specify the -E option (*preserve environment*) to start Wayland-compatible programs. nautilus is the program name of the GNOME file manager:

```
user$ sudo nautilus          (on X and in current Wayland versions)
user$ sudo -E nautilus       (gedit on old Wayland version)
```

10.3.1 sudo with Ubuntu

In Ubuntu and some other distributions, the root user is set up by default without a valid password. A root login is therefore impossible! In addition, su or ssh -l root do not work. The only way to execute admin commands is thus provided by sudo. The /etc/sudoers file contains only a few lines:

```
# Default configuration in /etc/sudoers on Ubuntu
Defaults   env_reset
Defaults   mail_badpass
Defaults   secure_path=
  "/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin"
Defaults   use_pty
root       ALL=(ALL) ALL
%admin     ALL=(ALL) ALL
%sudo      ALL=(ALL:ALL) ALL
```

`Defaults env_reset` causes all environment variables to be reset when the user changes. `Defaults mail_badpass` causes a warning email to be sent to the administrator after an incorrect login attempt. `Defaults secure_path` sets the contents of the `PATH` environment variable for `sudo` commands.

The fourth line grants `root` unrestricted access to all programs. This line is actually useless on Ubuntu because the `root` login is locked.

The last two lines are the most important: They allow all members of the `sudo` and `admin` groups to call all programs. Usually, on Ubuntu, only the first user, the one set up during the installation, is a member of the `sudo` group. Other users can also be assigned to this administrators group. In older Ubuntu versions, the `admin` group was used instead of the `sudo` group.

Special Treatment of Home

In the past, `sudo -s` did not use `/root` as the home directory, but `/home/<name>`. This had the advantage that configuration files like `/home/<name>/.emacs` applied equally to the user in question and to `root`. However, this configuration also caused conflicts with access rights (new configuration files created under `sudo` could not be modified in normal operation without `sudo`) and security problems. To avoid this, Ubuntu now also uses `/root` as the home directory for `sudo` commands.

10.3.2 sudo with Raspberry Pi OS

Even with the Raspberry Pi OS distribution optimized for the Raspberry Pi, there is no `root` login by default. The first user set up during setup has an unusually liberal `sudo` entry:

```
# /etc/sudoers
...
username ALL=(ALL) NOPASSWD: ALL
```

username may thus execute any command using sudo without a password. It is safer to prefix this line with the comment character #. username can then continue to use sudo because it belongs to the sudo group. However, the user must authenticate himself with his password.

10.3.3 sudo with Debian

On Debian, sudo is installed by default. Whether it is also active depends on how you carried out the installation:

- If you used the live image installer, or if you skipped entering the root password in the traditional installing program, then, as with Ubuntu, there is no root password. However, the user created during the installation is part of the sudo group.
- On the other hand, if you have specified a root password in a traditional installation, then sudo is not active (i.e., there is no user in the sudo group by default). To gain root privileges, you must run su -l and specify the root password.

10.3.4 sudo for RHEL and Fedora

In Fedora, as well as RHEL, you can set up a user during the installation process and make that user the administrator. The user is thus assigned to the wheel group. /etc/sudoers contains the following line for this group:

```
# in /etc/sudoers in Fedora
...
%wheel     ALL=(ALL) ALL
```

wheel group members must provide their own password to execute sudo commands. Separate from sudo, there is still the root user with its own password.

10.3.5 sudo with SUSE

sudo is also set up by default in SUSE distributions. However, the configuration differs significantly from that of Ubuntu, Fedora, and the like:

```
# Default configuration in /etc/sudoers with openSUSE
Defaults always_set_home
Defaults env_reset
Defaults env_keep = "LANG LC_ADDRESS LC_CTYPE ..."
Defaults targetpw    # sudo asks for the password of the target user
```

```
..
ALL  ALL=(ALL) ALL        # with the right password everyone
                          # is allowed to do everything
root ALL=(ALL) ALL
```

`Defaults targetpw` means that basically the password for the account in which the command is to be executed must be specified, usually the `root` password. The advantage of `sudo`—namely, that not all users with administrator tasks need to know the `root` password—is lost. Finally, the `ALL ALL=(ALL) ALL` line allows all users to execute any command, provided the correct password for the target account is known.

To run desktop programs using `sudo`, you must use the `-E` option—for example, `sudo -E console`.

10.4 Running Processes under a Different Identity (PolicyKit)

Concept

The basic idea of PolicyKit is to split programs into two components: One part contains the user interface and runs with ordinary user rights. The second part of the program, referred to as the *mechanism* in PolicyKit terminology, is responsible for system interventions and runs with `root` privileges.

This separation has the fundamental advantage that it is no longer necessary to run a huge program with `root` privileges, but only small parts of it. This reduces potential security risks. In addition, it is theoretically possible for different user interfaces (such as a GNOME program and a KDE program) to use a unified set of mechanisms.

The communication between the two components takes place via a so-called bus system, usually via D-Bus. Whether a certain mechanism may be executed or not is decided by functions of the PolicyKit library, which use a central rights database. Three criteria are considered for the decision:

- **Subject**
  Who or which user wants to make system changes?
- **Object**
  Which object should be changed (e.g., a file, a partition, or a network connection)?
- **Action**
  What should be done (e.g., mount a partition to the file system)?

User view

In many cases, the user doesn't notice anything about PolicyKit. For example, the default configuration in most current distributions allows the file manager to mount external disks in the file system. No further authentication is required for this; the process takes place automatically as soon as the data medium is connected.

You will come across another variant in various GNOME administration tools: here, a button marked with a padlock takes you to the authentication dialog. Only after specifying the `root` or user password can system changes be made.

Configuration

PolicyKit is configured in three places:

`/etc/polkit-1/*`	(global configuration, default settings)
`/usr/share/polkit-1/actions/*.policy`	(actions)
`/var/lib/polkit-1/*`	(rights)

There are distribution-specific peculiarities in the basic configuration. For example, `/etc/polkit-1/localauthority.conf.d/51-ubuntu-admin.conf` on Ubuntu systems gives admin privileges to all users of the `admin` and `sudo` groups.

pkexec

One component of the PolicyKit package is the aforementioned `pkexec` command, which is responsible for the actual execution of commands and analyzes the policy configuration in the process. For commands to be executed in the terminal, `pkexec` works without further configuration. Of course, it would be even easier to use `su` or `sudo` in this case:

```
user$ pkexec nano
```

However, for graphical programs not configured for `pkexec`, the call fails with the error message `cannot open display`:

```
user$ pkexec gnome-text-editor
Unable to init server ..., cannot open display
```

Custom policy files

Fortunately, it isn't difficult to set up your own `*.policy` file. The PolicyKit file of the `gparted` partition editor can serve as a sample. Once you have installed this program, you should take a look at the following file, which I reproduce here in a highly abbreviated form:

```
user$ cat /usr/share/polkit-1/actions/org.gnome.gparted.policy

<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE policyconfig PUBLIC ...>
<policyconfig>
    <vendor>The GParted Project</vendor>
```

```
    <vendor_url>https://gparted.org</vendor_url>
    <icon_name>gparted</icon_name>
    <action id="org.gnome.gparted">
        <description>Run GParted as root</description>
        <message>Authentication is required for GParted ...</message>
        ...
        <defaults>
            <allow_any>auth_admin</allow_any>
            <allow_inactive>auth_admin</allow_inactive>
            <allow_active>auth_admin</allow_active>
        </defaults>
        <annotate key="org.freedesktop.policykit.exec.path">
           /usr/bin/gparted
        </annotate>
        <annotate key="org.freedesktop.policykit.exec.allow_
        gui">true</annotate>
    </action>
</policyconfig>
```

You can create the equivalent `myown.editor.policy` file from a copy of `org.gnome.gparted.policy`. To do this, you only need to change a few lines and simply delete the translation texts in all conceivable languages:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE policyconfig ... (as above) >
<policyconfig>
    <action id="myown.editor">
        <description>Run GNOME Text Editor as root</description>
        <message>Authentication is required for GNOME Text</message>
        <defaults>
            <allow_any>auth_admin</allow_any>
            <allow_inactive>auth_admin</allow_inactive>
            <allow_active>auth_admin</allow_active>
        </defaults>
        <annotate key="org.freedesktop.policykit.exec.path">
           /usr/bin/gnome-text-editor
        </annotate>
        <annotate key="org.freedesktop.policykit.exec.allow_
        gui">true</annotate>
    </action>
</policyconfig>
```

The existence of this file subsequently allows you to start the editor in a terminal window by using `pkexec gedit`. Of course, a dialog for entering the `sudo` or `root` password still appears first.

10.5 System Processes (Daemons)

Background processes for system management are referred to as *daemons*. These processes are typically started as part of the init process when the computer boots up. If you're more familiar with Windows lingo, Linux daemons simply correspond to services. Table 10.1 very briefly describes the tasks of the most important daemons.

Where programs are described in this book, the relevant sections are indicated.

Process	Meaning
apache2	Web server (Chapter 24, Section 24.1)
atd	Starts other programs at specified times
avahi-daemon	Automatic network configuration (zeroconf)
bluetoothd	Bluetooth management
Cron	Starts other programs at specific times (Section 10.6)
cupsd	Printer spooler (Chapter 14, Section 14.11.2)
dbus-daemon	D-bus communication (Chapter 14, Section 14.8.5)
dhclient	DHCP client (Chapter 15, Section 15.4.2)
dhcpcd	Determines your own IP network address (Chapter 15, Section 15.2.4)
dhcpd	Assigns the IP network address to other computers
dovecot	IMAP and POP servers (Chapter 26, Section 26.5)
gdm	GNOME Login Manager (Chapter 17, Section 17.5)
httpd	Web server (e.g., apache; Chapter 24, Section 24.1)
lpd	Conventional printer spooler based on BSD-LPD
mdnsd	Automatic network configuration (zeroconf, Bonjour)
mysqld	MySQL or MariaDB database server (Chapter 25)
named	Domain name server
NetworkManager	NetworkManager (Chapter 15, Section 15.1)
nmbd	Name server for Windows/Samba (Chapter 27)

Table 10.1 Important System Processes

Process	Meaning
nscd	Cache for user and computer names (Chapter 14, Section 14.6.3)
postfix	Mail server for sending email (Chapter 26, Section 26.2)
rsyslogd	Logs system messages (Chapter 14, Section 14.12).
sddm	Simple Desktop Display Manager (KDE; Chapter 17, Section 17.5)
smartd	SMART hard disk monitoring (Chapter 18, Section 18.17)
smbd	File server for Windows/Samba (Chapter 27)
spamd	Detects spam mails (Chapter 26, Section 26.7)
sshd	Secure Shell Server (Chapter 23)
systemd-journald	Logs system messages (Chapter 14, Section 14.13)
systemd-udevd	Device management (Chapter 9, Section 9.9)
udevd	Device management (Chapter 9, Section 9.9)
vsftpd	FTP server

Table 10.1 Important System Processes (Cont.)

10.5.1 Kernel Threads

Besides ordinary server services like `httpd` (Apache), there are background processes, which are not real programs but subprocesses (threads) of the kernel. You can recognize these processes by the fact that `ps axu` puts their names in square brackets.

Some of these subprocesses are suffixed with a number that indicates the CPU. `kblockd/0` thus manages the block device buffer for the first CPU, `kblockd/1` the buffer for the second CPU, and so on.

Most kernel threads are related to low-level tasks of the operating system (memory management, process management, CPU control, etc.). They are usually started during system initialization. No special configuration is required for standard requirements.

The functions of the most important kernel threads are summarized in Table 10.2. If you want to use `ps` to display only kernel threads, you should run the command with the following options:

```
user$ ps -f -p 2 --ppid 2
```

Kernel Thread	Meaning
kacpid	ACPI functions
kblockd	Manages the block device buffer
kdevtmpfs	Manages the temporary /dev file system
khelperd	Loads or removes kernel modules for user programs
kthread	Manages threads
ksoftirqd	Hardware interrupt management
kswapd	Swapping
kworker	Execute kernel functions ("work")
md	RAID functions
migration	Determines which processes run on which CPU
scsi_eh	Manages SCSI errors and timeouts
watchdog	Monitors whether the system is still responding

Table 10.2 Selected Kernel Threads

10.5.2 Starting and Stopping System Services

systemd

The daemons listed in Table 10.1 are started via the init system, which I describe in detail in Chapter 20. Almost all common distributions use systemd as the init system.

For now, I offer only a brief summary of how to start or stop a system service manually, what you need to do to make the daemon start automatically at system startup, or how to avoid an automatic startup.

You'll need this information frequently, especially when setting up and configuring network services. Note that not only the command but also the service name may vary depending on the distribution. For example, the service for the Apache web server is called apache2 in Debian, Ubuntu, and SUSE, but httpd in Fedora and Red Hat:

```
root# systemctl start   name      (start once)
root# systemctl stop    name      (stop)
root# systemctl status  name      (get status)
root# systemctl restart name      (restart)
root# systemctl reload  name      (reload configuration)
root# systemctl enable  name      (start automatically in future)
root# systemctl disable name      (do not start in future)
```

On Debian and Ubuntu, newly installed server services will run immediately and automatically on every reboot in the future. For other distributions, once you have completed the configuration, you must run these two commands once:

```
root# systemctl start   name          (start now once)
root# systemctl enable  name          (also start in future)
root# systemctl enable --now name     (corresponds to start + enable)
```

10.6 Starting Processes Automatically (Cron)

If your computer suddenly—seemingly unexpectedly—starts to search the hard disk or SSD, sends you emails, and so on, then the cause is almost always the automatic launch of processes through the Cron daemon.

This program is started automatically by the init process when the computer boots. It becomes active once per minute, analyzes all `crontab` files, and launches the programs specified there. Cron is primarily used for maintenance tasks: to compress and archive logging files, to delete temporary files, to update directories, to perform backups, and so on.

The global configuration of Cron is done by the `/etc/crontab` file. In addition, users are allowed to define their own cron jobs in the user-specific files at `/var/spool/cron/[crontabs/]username`.

The permission for user-specific cron control can be set using two files: `/var/spool/cron/allow` and `/deny`.

If `allow` exists, only the users entered here may execute `cron` commands. If `deny` exists, the users entered here are excluded. If none of these files exist, it depends on the compilation of Cron whether any users other than `root` are allowed to use `cron`.

Cron, Where Are You?

Many Linux installations have Cron installed automatically—but this is not a matter of course (any more); modern distributions such as Fedora have completely switched to an alternative system for starting jobs periodically (see Section 10.7).

Cron is also sometimes missing in minimal installations, which are common in the server and virtualization area. To solve this problem, run `dnf install cronie` (Fedora/RHEL/SUSE) or `apt install cron` (Debian/Ubuntu)!

/etc/crontab

The /etc/crontab file or the files in /etc/cron.d contain line-by-line entries for the programs to run. The syntax looks as follows (see also Table 10.3):

```
# in /etc/crontab
min hour day month weekday user command
```

If an asterisk is entered in the first five fields instead of a number, this field is ignored. For example, 15 * * * * means that the command should always be executed 15 minutes after the full hour, in every hour, on every day, in every month, regardless of the day of the week. 29 0 * * 6 means that the command is executed every Saturday at 0:29.

For the time fields, the notation */*n* is also allowed. This means that the command will be executed every *n*th minute/hour/and so on. */15 * * * * would therefore mean that the command is executed every quarter of an hour (*n*:00, *n*:15, *n*:30, and *n*:45). To change the global Cron configuration, you can edit /etc/crontab or the files in /etc/cron.d/* directly in an editor.

Column	Meaning
min	Specifies in which minute (0–59) the program is supposed to be executed
hour	Indicates the hour (0–23)
day	Indicates the day in the month (1–31)
month	Indicates the month (1–12)
weekday	Indicates the day of the week (0–7; 0 and 7 both represent Sunday)
user	Specifies for which user the command is executed (often root)
command	Finally contains the command to be executed

Table 10.3 crontab Columns

Instead of the five time columns summarized in Table 10.3, the @ abbreviations listed in Table 10.4 can also be used. Another additional rule states that a minus sign at the beginning of the first column prevents syslog from logging the command execution. However, this is only allowed if the sixth column contains root.

Abbreviation	Code	Meaning
@reboot	-	Run after each reboot
@yearly	0 0 1 1 *	Run once a year

Table 10.4 crontab Interval Abbreviations to Replace First Five Columns

Abbreviation	Code	Meaning
@annualy	0 0 1 1 *	Like @yearly
@monthly	0 0 1 * *	Run once a month
@weekly	0 0 * * 0	Run once a week
@daily	0 0 * * *	Run once a day
@hourly	0 * * * *	Run once per hour

Table 10.4 crontab Interval Abbreviations to Replace First Five Columns (Cont.)

The crontab Syntax Requires a Line Break after the Last Line

Make sure that all Cron configuration files end with a line break: otherwise the last line will be ignored!

Crontab example

The following lines provide a few examples of entries in `/etc/crontab`:

```
# Run backup every night at 1:45
45 1 * * * root /myscripts/backup-site
# Cleanup on 1st day of each month at 6:00
0  6 1 * * root /myscripts/cleanup
# Log network connection every 10 minutes
*/10 * * * * root /myscripts/log-ping-result
```

Error email

When a Cron job returns output or error messages, they are automatically sent to `root@localhost`. By setting the `MAILTO` variable in `crontab`, you can also set a different email address. Note, however, that sending nonlocal emails requires a configured email server.

User-specific crontab files

The `/var/spool/cron/[crontabs/]user` files have the same format as `crontab`. The only difference is that the `user` column is missing because this information already comes from the name of the crontab file. To change user-specific Cron entries, you should use the `crontab -e` command. Run `export EDITOR=emacs` beforehand if you do not want to use `vi`. `man cron` and `man crontab` contain more detailed information.

10.6.1 /etc/cron.hourly, .daily, .weekly, and .monthly

All common distributions have the following four directories set up:

`/etc/cron.hourly/` (run contained scripts hourly)
`/etc/cron.daily/` (run contained scripts daily)

```
/etc/cron.weekly/      (run contained scripts weekly)
/etc/cron.monthly/     (run contained scripts monthly)
```

Thus, you can store your own scripts in these directories, but don't forget to set the *execute* bit (`chmod a+x file`)! The scripts are then executed once per hour, day, week, or month with `root` privileges. Using the `/etc/cron.xxx-ly` directories prevents you from thinking about the correct crontab syntax. In addition, compared to conventional crontab entries, these directories have the advantage that scripts defined in this way are executed reliably even if the computer isn't running all the time.

The distributions differ significantly in the *way* the scripts contained in the four directories are executed. Most distributions require the Anacron program described in the next section to be installed, and Anacron then takes care of the execution.

However, on Ubuntu, running the `/etc/cron.xxx-ly` scripts works without Anacron. This is accomplished by the following crontab configuration:

```
# /etc/crontab on Ubuntu
SHELL=/bin/sh
PATH=/usr/local/sbin:/usr/local/bin:/sbin:/bin:/usr/sbin:/usr/bin
17 *    * * *   root    cd / && run-parts --report /etc/cron.hourly
25 6    * * *   root    test -x /usr/sbin/anacron || \
  ( cd / && run-parts --report /etc/cron.daily )
47 6    * * 7   root    test -x /usr/sbin/anacron || \
  ( cd / && run-parts --report /etc/cron.weekly )
52 6    1 * *   root    test -x /usr/sbin/anacron || \
  ( cd / && run-parts --report /etc/cron.monthly )
```

In plain language, this means that Cron does the following:

- Runs all script files of the `/etc/cron.hourly` directory 17 minutes after every full hour
- Runs all script files of the `/etc/cron.daily` directory at 6:25 a.m. every day
- Runs all script files of the `/etc/cron.weekly` directory every Sunday at 6:47 a.m.
- Runs all script files of the `/etc/cron.monthly` directory on every first day of the month at 6:52 a.m.

If Anacron is installed, `test -x` returns a positive result and `run-parts` is not called. This isn't even necessary because now Anacron takes control of the scripts in the four `/etc/cron.xxx-ly` directories.

Rules for File Names in /etc/cron.daily, -.weekly, and -.monthly

On Debian and Ubuntu, the file names of custom scripts in `cron.daily`, `cron.weekly`, and `cron.monthly` may only consist of numbers, letters, hyphens, and underscores! As soon as the filename contains even one dot, the script will be ignored by `run-parts`! These rules are intended to prevent backup files that are created when a script file is modified from being executed as well.

If you want to check which scripts from `run.daily` are executed daily, you need to run `run-parts --test /etc/cron.daily`.

SUSE distributions are also independent of Anacron. Here, the `run-crons` script takes care of the `/etc/cron.xxx-ly` directories. This script is executed every quarter of an hour:

```
# /etc/crontab on SUSE
-*/15 * * * *   root  test -x /usr/lib/cron/run-crons && \
                      /usr/lib/cron/run-crons >/dev/null 2>&1
```

`run-crons` isn't quite as pedantic as Ubuntu's `run-part` when it comes to checking file names. Only those scripts whose names look like backup files will not be executed.

10.6.2 Anacron

Cron assumes that the computer is running all the time, as is the case with servers. The situation is quite different for notebook computers or desktop PCs, which are frequently switched on and off again: Anacron was developed to ensure that tasks to be performed daily, weekly, or monthly are completed even in that case.

Anacron stores the execution time in files of the `/var/spool/anacron` directory. This prevents a script intended for daily execution from being executed twice on the same day.

Anacron is controlled by `/etc/anacrontab`. In most distributions, the default configuration is as follows:

```
# File /etc/anacrontab (Debian, Ubuntu)
1        5   cron.daily       nice run-parts /etc/cron.daily
7        10  cron.weekly      nice run-parts /etc/cron.weekly
@monthly 15  cron.monthly     nice run-parts /etc/cron.monthly
```

Here, the first column indicates how often the task should be done (1 = daily; 7 = once a week). The second column gives a delay time in minutes

that Anacron allows to elapse before executing `run-parts`. This time period is to prevent daily, weekly, and monthly tasks from being completed at the same time.

On RHEL, the `/etc/cron.hourly/0anacron` script, which is executed once an hour by Cron, is responsible for Anacron.

Random delay of Cron jobs

If many Cron jobs start at the same time, this may cause heavy CPU or I/O load for the system. One option is to just consistently set up the jobs so that this doesn't occur. Another option is provided by the `RANDOM_DELAY` variable, which can be defined in `anacrontab`. This variable specifies a random delay time in minutes. `RANDOM_DELAY=20` causes each job executed by Anacron to be delayed by a random period of 0 to 20 minutes.

10.7 Starting Processes Automatically (systemd Timer)

Almost all distributions use systemd as an init system—that is, to automatically start and stop processes when the computer boots or is shut down. A detailed introduction of systemd in this function follows in Chapter 20, Section 20.1. However, systemd can also take care of the regular execution of processes, similar to Cron. This aspect of systemd is the focus here.

Most distributions make very sparing use of the systemd timer functions. In this respect, systemd should not yet be seen as a replacement for Cron, but rather as a supplement. The systemd timer does, however, have the potential to replace Cron in the longer term. A pioneer in this respect is, as usual, Fedora, which has already been completely converted to systemd timers. Cron is no longer installed by default (but is of course still available as a package).

Systemd timers

The time-controlled execution of programs is controlled in systemd by timer unit files—that is, files with the extension `.timer`. An overview of active timers is provided by `systemctl`, although the following output is abbreviated to save space:

```
user$ systemctl list-timers
NEXT                LEFT      LAST                UNIT
2024-04-13 11:01  3min 14s  2024-04-13 09:41  dnf-makecache.timer
2024-04-14 00:00  13h       2024-04-13 07:21  logrotate.timer
2024-04-14 00:00  13h       2024-04-13 07:21  plocate-updatedb.timer
2024-04-14 00:00  13h       2024-04-13 07:21  unbound-anchor.timer
2024-04-14 09:36  22h       2024-04-13 09:36  systemd-tmpfiles-clean.timer
2024-04-18 01:00  4 days    2024-04-13 07:21  raid-check.timer
```

.timer-Syntax The `.timer` files are in an easy-to-understand text format. The following lines show the `dnf-makecache.timer` file, which is responsible for the daily start of the `dnf makecache` command. It updates the package database:

```
# file /usr/lib/systemd/system/dnf-makecache.timer
[Unit]
Description=dnf makecache --timer
ConditionKernelCommandLine=!rd.live.image
ConditionPathExists=!/run/ostree-booted
Wants=network-online.target

[Timer]
OnBootSec=10min
OnUnitInactiveSec=1h
RandomizedDelaySec=60m
Unit=dnf-makecache.service

[Install]
WantedBy=timers.target
```

`OnBootSec=10min` causes the process to be executed for the first time ten minutes after a reboot of the computer. `OnUnitInactiveSec=1h` means that the service should be restarted one hour after the end of the last run. If the time is given without a unit, seconds are meant. The syntax of the time specifications is documented in `man systemd.time`. For example, two hours, 15 minutes or two weeks or four months is permitted.

`RandomizedDelaySec=60min` increases the time between two starts by a random time between 0 and 60 minutes. On average, `dnf makecache` thus runs every 90 minutes.

Instead of these rather free defaults, you can also enter the time specifications in absolute terms in the `[Timer]` section. The following settings will cause the task to be completed every day at 8:15 a.m. `Persistent=true` causes missed jobs to be caught up on immediately the next time the computer boots:

```
# in a .timer file
[Timer]
OnCalendar=8:15
Persistent=true
```

For absolute time specifications, systemd provides a complex syntax, which is documented in detail in `man systemd.time` in the Calendar Events section. For example, `OnCalendar=Sun 2023-*-* 17:15` causes a job to run every Sunday at 5:15 p.m. in the year 2023. Systemd attempts to run set up

timer jobs in a one-minute window after the scheduled time by default. If you want systemd to adhere more precisely to your specifications, you can enter a shorter time interval in the `[Timer]` section using `AccuracySec`—for example, from `1s` (one second) to `1us` (theoretically one millionth of a second, but in reality the start will not always be quite as precise).

.service file

Each `name.timer` file must be matched by a corresponding `name.service` file that contains details of the process to be executed. For the `dnf-makecache` example, the `.service` file looks as follows:

```
# file /usr/lib/systemd/system/dnf-makecache.service
[Unit]
Description=dnf makecache
ConditionPathExists=!/run/ostree-booted
After=network-online.target

[Service]
Type=oneshot
Nice=19
IOSchedulingClass=2
IOSchedulingPriority=7
Environment="ABRT_IGNORE_PYTHON=1"
ExecStart=/usr/bin/dnf makecache --timer
```

Start/stop

Newly set up timers must be started for the first time via `systemctl start` and activated permanently (i.e., until after the next reboot) by `systemctl enable`. Similarly, you can end the call using `systemctl stop` and `systemctl disable`. For changes to take effect in already active timers, the easiest way is to run the following command:

```
root# systemctl reenable --now name.timer
```

After setting up new timer jobs, you should always use `systemctl list-timers` to make sure that systemd has correctly interpreted your `.timer` specifications.

Example

The following example shows how you can use systemd to run a simple script that checks the network connection every 10 minutes. By `ping`ing three times, the `ping-kofler` script tests whether and how fast the `kofler.info` server responds to network requests. The current time and the `ping` output are logged in `/var/log/ping-kofler.log`. `&>>` causes both the standard output and any errors that occur to be added to an existing file. This operator requires `bash` version 4. Do not forget to make the script executable by using `chmod a+x`.

```
#!/bin/bash
# file /usr/local/bin/ping-kofler
date &>> /var/log/ping-kofler.log
ping -c 3 kofler.info &>> /var/log/ping-kofler.log
```

The following `.service` file is responsible for calling the script:

```
# file /etc/system/system/ping-kofler.service
[Unit]
Description=simple ping test

[Service]
ExecStart=/usr/local/bin/ping-kofler
```

The settings for an automatic startup are located in the following `.timer` file. The script is supposed to be run for the first time 30 seconds after `systemctl start` and then every 10 minutes. The `[Install]` section is required to enable the timer permanently via `systemctl enable`:

```
# file /etc/systemd/system/ping-kofler.timer
[Unit]
Description=ping-kofler timer

[Timer]
OnActiveSec=30s
OnUnitActiveSec=10m

[Install]
WantedBy=basic.target
```

To start the script execution and set it up permanently, another `systemctl` command is needed:

```
root# systemctl enable --now ping-kofler.timer
```

Cron/Anacron versus Systemd

The example immediately makes it clear that setting up a systemd timer involves much more effort than the one `crontab` line that would be required for a periodic Cron execution.

The advantages of systemd are that systemd jobs can be controlled more precisely (own environment, see `man systemd.exec`, `cgroups` rules, etc.) and that their calls are logged in the systemd journal.

But there are also disadvantages: for example, an email is not automatically sent in case of an error.

Documentation

For more information about the systemd timer configuration, see `man systemd.timer` and the following pages:

- *https://wiki.archlinux.org/title/Systemd/Timers*
- *https://blog.thalheim.io/2013/06/09/use-systemd-as-a-cron-replacement*

Chapter 11
Network Tools

This chapter introduces commands for using, controlling, and analyzing essential network services. You will learn how to log into another computer on the network using `ssh` and how to transfer files by using `wget`. The Lynx and Mutt programs even enable you to visit websites and read and compose mails in text mode.

Note that this chapter is only an initial starting point. Descriptions of other network commands follow in the context of other topics:

- Commands for network configuration in Chapter 15
- SSH variants and alternatives in Chapter 23
- Access to Windows and Samba network directories in Chapter 27, Section 27.7
- Network security and firewalls in Chapter 29

11.1 Determining the Network Status

This section provides an overview of commands for testing the basic functions of the network. More information about the commands presented here follows in Chapter 15, Section 15.3.1.

Determining the IP address

Without any additional options, the `hostname` command displays the host name as expected. What is less well known is that `hostname -I` (this is an uppercase i—not a one, not even an L) shows the IP addresses under which the computer is visible to the outside world or within the local network. In the following example, the computer is assigned one IPv4 and two IPv6 addresses:

```
user$ hostname -I 192.168.178.50
2001:871:264:5779:82fd:84b3:6e09:60db
2001:871:264:5779:fcd1:5235:d5d6:20dd
```

The result of `hostname -I` is a simplification in that every Linux computer has *multiple* IP addresses even with simple setups. If you use the computer for program development and use virtual machines, Docker, or the like, then a dozen or more IP addresses are quickly in play. The `ip` command (discussed ahead) lists all IP addresses—but the really relevant information is often difficult to find in the resulting jumble of numbers. `hostname -I` attempts to filter out the really relevant addresses.

Determining network interfaces

The `ip link` command returns a list of all network interfaces. The following result is from a different `host` than that of `hostname -I`:

```
user$ ip link
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue ...
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
2: enp0s31f6: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 ...
    link/ether 48:2a:e3:16:aa:24 brd ff:ff:ff:ff:ff:ff
3: wlp0s20f3: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 ...
    link/ether 34:e1:2d:e7:c1:c4 brd ff:ff:ff:ff:ff:ff
4: virbr0: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 ...
    link/ether 52:54:00:90:3c:0b brd ff:ff:ff:ff:ff:ff
5: docker0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 ...
...
25: vnet8: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 ...
```

In this result, `lo` denotes the loopback interface, which is a virtual internal network interface that is always assigned the localhost addresses 127.0.0.1 (IPv4) and ::1 (IPv6). Local programs on the computer can communicate with each other via these addresses.

Typical interfaces are `eth<n>` or `enp<n>s<m>` (Ethernet) and `wlan<n>` or `wlp<xxx>` (Wi-Fi). Most common distributions include the internal bus number in the name of the Ethernet interface, such as `enp0s1`. On PCs and notebook computers with only one Ethernet interface, the name seems cumbersome. But the bus-specific numbering ensures that on servers with many network interfaces, the numbering doesn't change even when more network adapters are added. So in this example, `enp0s31f6` and `wlp0s20f3` are names for two real interfaces.

The first twelve-digit number combination after `link` indicates the so-called MAC address, which uniquely designates each interface (e.g., `48:2a:e3:16:aa:24` for the Ethernet interface).

All other interfaces from the preceding example exist only virtually; that is, they are generated by software. They enable network traffic between Docker containers and virtual machines.

Details of the IP addresses assigned to an interface are provided via `ip addr show <name>`. In its short form, `ip addr`, the command shows details of all interfaces, but this often overshoots the mark. The following listing shows that the `enp1s0` interface from the `hostname` example has one IPv4 address and three IPv6 addresses assigned to it. The address beginning with `fe80` is only intended for internal communication on the local network (for details on this and on IPv6 configuration, see Chapter 15):

```
root# ip addr show enp1s0
2: enp1s0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc ...
    link/ether 52:54:00:6e:38:8e brd ff:ff:ff:ff:ff:ff
    inet 192.168.178.50/24 brd 192.168.178.255 scope global ...
       valid_lft 861765sec preferred_lft 861765sec
    inet6 2001:871:264:5779:82fd:84b3:6e09:60db/64 scope ...
       valid_lft 7097sec preferred_lft 2863sec
    inet6 2001:871:264:5779:fcd1:5235:d5d6:20dd/64 scope ...
       valid_lft 7097sec preferred_lft 2863sec
    inet6 fe80::f126:174:824e:22cc/64 scope link noprefixroute
       valid_lft forever preferred_lft forever
```

Testing the accessibility of localhost

`ping` sends a small network packet to the specified address once per second. If there is a computer there, it sends a response, unless a firewall prevents this. `ping` keeps running until it is terminated via Ctrl+C. `ping localhost` checks whether the loopback interface and thus the elementary network functions of its own computer are working:

```
user$ ping localhost
PING localhost (127.0.0.1): 56 data bytes
64 bytes from 127.0.0.1: icmp_seq=0 ttl=255 time=0.152 ms
64 bytes from 127.0.0.1: icmp_seq=1 ttl=255 time=0.114 ms
...
```

Testing the accessibility of the local network

By passing the IP number of another computer in the local network to `ping` instead of `localhost`, you can test whether the local network is working. `-c 2` means that ping does not run endlessly, but that it will terminate after two packets:

```
user$ ping -c 2 192.168.178.1
PING 192.168.178.1 (192.168.178.1) 56(84) bytes of data.
64 bytes from 192.168.178.1: icmp_seq=1 ttl=64 time=1.33 ms
64 bytes from 192.168.178.1: icmp_seq=2 ttl=64 time=0.817 ms
...
```

If there is a name server on the local network that assigns IP addresses to host names, or if the `/etc/hosts` file takes care of this task, you can specify the host name instead of the IP number when `pinging`:

```
user$ ping -c 2 raspberrypi
PING raspberrypi(raspberrypi.fritz.box (2001:...:6973)) 56 data bytes
64 bytes from raspberrypi.fritz.box (2001:...:6973): icmp_seq=1 ...
time=0.525 ms
64 bytes from raspberrypi.fritz.box (2001:...:6973): icmp_seq=2 ...
time=0.606 ms
...
```

If both IPv4 and IPv6 addresses are available, `ping` usually opts for IPv6. If you want to explicitly test one protocol or the other, you must run `ping -4` or `ping -6`. On Debian and Ubuntu, there are even separate commands available for this (`ping4` and `ping6`).

Testing the internet access

Next, you can test if the connection to the internet succeeds. The following command tests two aspects of the network configuration simultaneously—the reachability of the name server and the function of the gateway:

```
user$ ping -c 2 google.com
PING google.com(fra24s04-in-x0e.1e100.net (2a00:...::200e)) 56 data bytes
64 bytes from fra24s04-in-x0e.1e100.net (2a00:...::200e): ... time=19.4 ms
64 bytes from fra24s04-in-x0e.1e100.net (2a00:...::200e): ... time=20.1 ms
...
```

If this doesn't work, several causes are conceivable:

- Google's server may be unreachable right now (this is very unlikely), or the server may have disabled the `ping` response for security reasons. Try another well-known internet address.
- The name server is responsible for determining the IP address for `google.com`. If you receive the error message `unknown host google.com`, there are problems with the name server. Verify that `/etc/resolv.conf` contains the address of a name server.
- The gateway is responsible for forwarding IP packets from the local network to the internet. If this doesn't work, you'll receive the error message `connect: Network is unreachable`. You can check the gateway configuration using `ip route` and `ip -6 route`. The command usually returns multiple lines. The gateway address is located in the line that starts with `default`:

  ```
  user$ ip route
  default via 10.0.0.138 dev eth0
  10.0.0.0/24 dev eth0 proto kernel scope link src 10.0.0.42
  ```

- If you've set up your own computer as a gateway in a local network, it's possible that you've forgotten the masquerading function. In that case, internet access for the entire local network wouldn't work.

Tracing the route of IP packets

You can use `traceroute` to determine which route a network packet takes from your computer to another computer and how many milliseconds the runtime is to the respective intermediate station. By default, the command makes three attempts and therefore returns three separate times accordingly. The command won't work if there is a firewall on one of the intermediate stations that blocks UDP port 33434 used by `traceroute`. In this case, `traceroute` returns only three stars for this and all other stations.

The following lines show the route from my work computer to `google.com`. Line 1 specifies the IP address of my router, line 2 the gateway of the internet provider:

```
user$ traceroute google.com
traceroute -I google.com
traceroute to google.com (142.250.186.46), 30 hops max, 60 byte packets
 1  fritz.box (192.168.178.1)  0.829 ms  1.050 ms  1.423 ms
 2  62-46-79-254.adsl.highway.telekom.at (62.46.79.254)  6.399 ms  ...
 3  195.3.74.153 (195.3.74.153)  10.155 ms  10.279 ms  11.405 ms
 ...
 8  fra24s04-in-f14.1e100.net (142.250.186.46)  19.852 ms  18.986 ms
19.081 ms
```

Bypassing Firewalls

Sometimes security settings and firewalls hinder the work of `traceroute`. Instead of IP addresses, the command then displays only * * *. In such cases, you can try using other methods to trace the route of packets using the `-T` or `-I` options. Both options require `root` privileges.

gnome-nettool

In GNOME, you can find most of the information listed so far (and much more) quite comfortably using the `gnome-nettool` program (see Figure 11.1). Some distributions require you to install the package of the same name first.

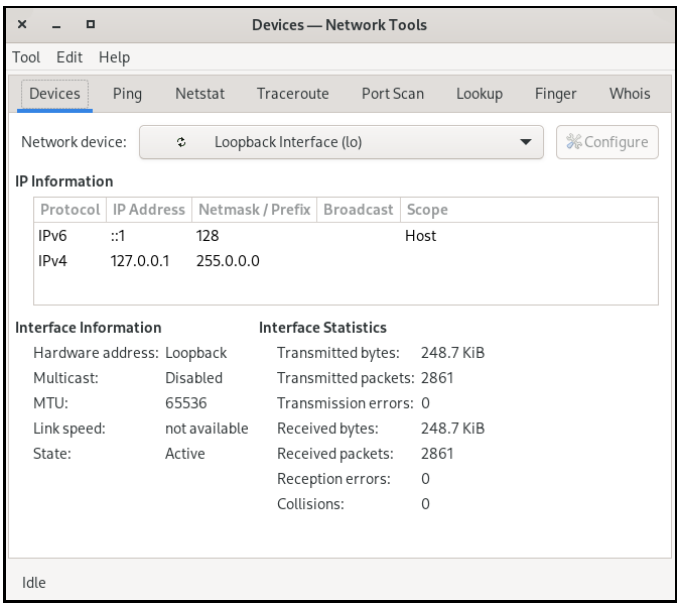


Figure 11.1 Network Diagnostics in GNOME

mtr The `mtr` command periodically sends network packets to the specified host and analyzes the responses. The result list combines data from `ping` and `traceroute`:

```
user$ mtr -c 10 -r google.com
Start: 2024-04-16T14:17:47+0200
  HOST: u2104         Loss%  Snt  Last   Avg  Best  Wrst StDev
  1.|-- fritz.box      0.0%   10   1.4   1.3   1.0   1.9   0.3
  2.|-- dynamic-2ky... 0.0%   10   5.9   5.7   5.1   6.3   0.4
  3.|-- 2001:850:1:... 0.0%   10   8.8   8.5   7.6   9.0   0.4
 ...
 11.|-- 2001:4860:0   10.0%   10  19.7  19.9  19.6  20.1   0.2
 12.|-- fra24s07-in... 0.0%   10  19.8  20.0  19.6  20.3   0.2
```

11.2 Working on Other Computers (SSH)

The `ssh` command allows you to work on another computer in text mode as if it were in front of you. This would also be possible by using `telnet` and `rlogin`—but these two commands are considered obsolete for security reasons as they require you to transmit login information, including the password, unencrypted.

The basic requirement for using `ssh` is that an SSH server is running on the second computer—that is, the `sshd` program. In some Linux distributions this is the case by default; in others, the program (mostly as the `openssh-server` package) has to be installed and/or enabled first (`systemctl start sshd` and `systemctl enable sshd`). If firewalls are running on the computers, they must not block port 22.

> **Setting Up Your Own SSH Server**
>
> Information on installing, configuring, and securing an SSH server follows in Chapter 23. There you will also learn how to secure the SSH server.

Ordinary shell session If you use the `uranus` machine and want to start a shell session on the `mars` machine on the local network, you need to run the following command to establish the connection:

```
user@uranus$ ssh mars
user@mars's password: ********
```

When you connect to a new computer for the first time, a warning appears according to the following pattern:

```
The authenticity of host 'mars (192.168.0.10)' can't be established.
RSA1 key fingerprint is 1e:0e:15:ad:6f:64:88:60:ec:21:f1:4b:b7:68:f4:32.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added 'mars,192.168.0.10' (RSA1) to the list
  of known hosts.
```

This means that `ssh` isn't sure whether it can trust the computer `mars` with the IP address 192.168.0.10. It could be that a foreign computer is pretending to be `mars`. If you answer "yes" to the prompt, `ssh` stores the name, address, and RSA fingerprint (a code to uniquely identify the partner host) in `~/.ssh/known_hosts`.

If you want to work on `mars` under a different login name than on `uranus` (e.g., as `root`), you can use the notation `username@hostname`:

```
user@uranus$ ssh root@mars
root@mars's password: *********
```

SSH Authentication with Keys

Authentication using a key is much more secure than logging in with a password. The procedure is described in detail in Chapter 23, Section 23.4. The use of keys also allows SSH-based commands and scripts to be executed automatically via script.

Security warning after key change

If you perform a Linux reinstall on a machine while keeping the previously used hostname, a dramatic security warning appears the next time you log in:

```
user$ ssh pi@raspberrypi
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@       WARNING: POSSIBLE DNS SPOOFING DETECTED!          @
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
The ECDSA host key for raspberrypi has changed,
and the key for the corresponding IP address 2001:...:60db
is unknown. This could either mean that
DNS SPOOFING is happening or the IP address for the host
and its host key have changed at the same time.
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@    WARNING: REMOTE HOST IDENTIFICATION HAS CHANGED!     @
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
IT IS POSSIBLE THAT SOMEONE IS DOING SOMETHING NASTY!
```

```
Someone could be eavesdropping on you right now (man-in-the-middle
attack)!

It is also possible that a host key has just been changed.

The fingerprint for the ECDSA key sent by the remote host is
SHA256:HX5LOrT526ZqWrKKbWEvTAyIVGq+6XNf+Oc+76efqGY.

Please contact your system administrator.

Add correct host key in /home/kofler/.ssh/known_hosts to get rid of
this message.

Offending ECDSA key in /home/kofler/.ssh/known_hosts:235
  remove with:
  ssh-keygen -f "/home/kofler/.ssh/known_hosts" -R "raspberrypi"
ECDSA host key for raspberrypi has changed and you have requested
strict checking.

Host key verification failed.
```

The `ssh` command is suspicious that the SSH key used by the `raspberrypi` computer has changed since the last connection was established and it fears that you could be the victim of a hacking attack. If it is a foreign server, you will now have to go to the trouble of finding out why the SSH key change occurred. In most cases, you or someone else has performed a fresh installation on the computer. At the first start, the SSH server has generated a new key. In this case, the solution is simple. You delete the previous key from `.ssh/known_hosts` (either using the following command or in a text editor) and then repeat the connection setup:

```
user$ ssh-keygen -f "/home/kofler/.ssh/known_hosts" -R "raspberrypi"
user$ ssh pi@raspberrypi
The authenticity of host 'raspberrypi
(2001:871:264:5779:82fd:84b3:6e09:60db)'
can't be established.

ECDSA key fingerprint is
SHA256:HX5LOrT526ZqWrKKbWEvTAyIVGq+6XNf+Oc+76efqGY.

Are you sure you want to continue connecting (yes/no)? yes
```

Running commands

Instead of using ssh interactively, you can run a command directly on the remote computer. The command and its parameters are simply passed as additional parameters to ssh. ssh ends after this command:

```
user@uranus$ ssh mars command options
user@mars's password: *********
```

There are far-reaching possibilities that arise from this seemingly trivial function. For example, you can now run tar on the remote machine, redirect the archive it creates to standard output (enter a dash - after the -f option, i.e., -f -), and use the standard output with | as input for a second tar command that runs locally. This allows you to copy an entire directory tree securely via SSH.

The following command shows how I copy the entire */var/www* directory tree of my kofler.info web server to the local *~/bak* directory. The command assumes that all files in /var/www can be read by the user username:

```
user$ ssh username@kofler.info tar -cf - /var/www | tar -xC ~/bak/ -f -
username@kofler.info's password: ******
```

Running multiple commands (scripts)

If you want to execute multiple commands in one script via SSH, it's best to use the Heredoc syntax described in Chapter 7, Section 7.11.8:

```
#!/bin/bash
pw=newPasswordForUsername
...
ssh -T root@host <<ENDSSH
echo username:$pw | chpasswd
rm -f /etc/file1
cp /root/file2 /userxy/file3
ENDSSH
...
```

This causes ssh to execute all commands until the line marked ENDSSH is reached in the script. The -T option prevents SSH from trying to open a pseudoterminal. This is not desirable here because the command execution isn't supposed to be interactive.

Preventing a query at first connection setup

The first time you establish an SSH connection to a new host, ssh asks if you trust the host. Basically, this query is useful. However, if you want to use ssh to perform automated work on multiple hosts or virtual machines, the query interferes. The solution in such cases is the -o Strict\-HostKeyChecking=no option. You can also set this and other options globally in /etc/ssh/ssh_config or individually for a user in .ssh/config. But do not confuse

/etc/ssh/ssh_config with /etc/ssh/sshd_config! The first file contains SSH client options; the second file contains options for the SSH server.

SSH and X In an SSH connection that you have initiated using ssh -X, you can also run graphics programs. The -X option is required for ssh to take care of setting the DISPLAY variables correctly. Thanks to XWayland, ssh -X for X programs also works in combination with Wayland.

```
user@localhost$ ssh -X otheruser@otherhost
otheruser@otherhost$ firefox   (Firefox runs externally, but is displayed locally)
```

11.2.1 Copying Files Securely Using scp

To copy a file via SSH over the network, you can use the scp command. The syntax looks as follows:

```
user$ scp [[user1@]host1:]filename1 [[user2@]host2:][filename2]
user2@host2's password: ******
```

This transfers the filename1 file from host1 to host2, where it's stored in the filename2 file. Some notes on the many optional components of the copy statement:

- host1 and host2 do not need to be specified if the local host is meant.
- user1 need not be specified if the active user is meant.
- user2 doesn't need to be specified if the current user name of host1 or user1 is to be used on host2.
- filename1 can also be a directory. You must then specify the -r option so that the entire directory with all subdirectories gets transferred.
- filename2 need not be specified if the filename is to remain unchanged. The file is then copied to the home directory of user2.

  Instead of filename2, the target directory can also be specified, using ~ for the home directory of user2 as usual.

Let's assume that user gladys works on computer uranus. She wants to transfer the abc.txt file to the ~/efg directory on computer mars. The scp command looks as follows:

```
gladys@uranus$ scp abc.txt mars:~/efg/
gladys@mars's password: ******
```

If you want to specify an IPv6 address in the scp command, you must enclose it in square brackets. Otherwise, scp will be confused by the many colons:

```
user$ scp kofler@[2001:1234:5678::1]:file.txt .
```

`scp` can also recursively transfer entire directory trees:

```
user$ scp -rp srcdir user@desthost:/destdir
```

Instead of copying files using `scp`, you can also use `rsync` (see Chapter 28, Section 28.6). The SSH-based SFTP protocol is used for this purpose.

11.2.2 SSH Tunnel

One possible SSH application for advanced Linux users is tunneling. While the resulting tunnels aren't suitable for transporting cars or trains, they do enable the transmission of all IP packets directed to a specific port. SSH tunnels thus provide a secure way to transfer IP packets between two computers—even if there is a firewall between the two computers that actually blocks the port. For an introduction to the world of IP packets and an explanation of the term *port*, see Chapter 29.

If tunneling is done from the client computer, the `-L localport:localhost:remoteport` option is used. For example, the following command causes port 3306 of host `mars` to be accessible through port 3307 of the local host. The command will start an SSH session at the same time, but you can prevent this by using `-N` (if you only need the tunnel, but not a shell). If the login to `mars` is supposed to be under a different name, you must specify the login name as usual by `-l name` or by `name@remotehost`:

```
user@uranus$ ssh -L 3307:localhost:3306 username@mars
user@mars's password: ******
```

The tunnel remains open until the SSH session is terminated via Ctrl+D. If you started `ssh` with the `-N` option, the program must be stopped using Ctrl+C. The usual MySQL port is 3306. You can now access the MySQL server running on `mars` from the `uranus` computer via its port 3307.

For the `mysql` command, you must specify port 3307 and the host name 127.0.0.1 so that the SSH tunnel is actually used. By default, `mysql` makes local connections through a socket file:

```
user@uranus$ mysql -u mysqllogin -P 3307 -h 127.0.0.1 -p
Enter password: *******
```

For the MySQL login to work, two requirements must be met:

- First, the MySQL server on computer `mars` must accept IP connections in general. The MySQL server can also be configured to allow connections only through a socket file for security reasons. Then a tunnel does not help, because a tunnel can only connect ports.

- Second, the MySQL server must accept the login name and host name combination. The host name used is the name of the computer to which `ssh` has established the tunnel: in this case, `mars` or `mars.sol` if the domain is `sol`.

11.2.3 SSH File System

You can integrate the file system of an external computer into the local directory tree using the `sshfs` command, which is included in the package of the same name in many distributions. This can make it easier to perform backups, for example:

```
root# mkdir /media/ext-host
root# sshfs user@hostname /media/ext-host
root# ...
root# umount /media/ext-host
```

Note, however, that you will mostly get lower throughput in the SSH file system than with Samba or NFS because of the encryption of all data. For this reason, the SSH file system is only suitable for use in local networks to a limited extent. I have also found that `sshfs` reacts allergically to brief network outages and then freezes. I have therefore abandoned the use of this otherwise useful file system.

11.2.4 telnet

`telnet` was a predecessor of `ssh`. Because `telnet` doesn't encrypt any data, the command should never be used to work on external computers. Current Linux distributions don't allow this by default anyway, but you can always find routers that allow this kind of communication.

The reason I present `telnet` to you here at all is that `telnet` is good for checking whether a network service is running on an external computer on a certain port and waiting for a connection to be established. For example, you can use `telnet` to make sure that the mail server you set up earlier is actually running. To do this, you need to pass `telnet` the name or IP address of the server and the port number:

```
user$ telnet kofler.info 25
Trying 5.9.22.29...

Connected to kofler.info.

Escape character is '^]'.
220 kofler.info ESMTP Postfix (Ubuntu)
```

```
helo kofler.info
250 kofler.info
^]      (terminate connection using Ctrl + ])
```

11.3 Transferring Files (FTP and Others)

Basic principles

FTP stands for *File Transfer Protocol* and refers to a rather old method of transferring files through a network. FTP owes its great popularity to *anonymous* FTP: many large internet servers offer all users access to FTP archives. This access is not blocked by a password (unlike the other FTP).

A major drawback of FTP is that during the login process the username and password are transmitted unencrypted. A secure alternative is SFTP (Secure FTP) based on SSH (see Chapter 23). HTTP, the protocol used to transfer web pages, is also often used as an alternative to FTP.

This chapter is only about using FTP—that is, from the client point of view. For FTP to work, an FTP server must be running on the remote site.

FTP command

The ancestor of all FTP clients is the interactive text command `ftp`. Because it typically transfers files from or to the current directory, you should use `cd` to change to the relevant working directory before starting `ftp`. The FTP session can then be initiated using the command `ftp user@ftpservername` or simply `ftp ftpservername`. If you want to use anonymous FTP, you need to enter `anonymous` as the user name.

Once the connection has been established and the password entered, you're ready to go: You can use the `cd`, `pwd`, and `ls` commands, which have the same meaning as in Linux, to move through the directories of the FTP archive. To transfer a file from the FTP archive to the current directory of your computer, you can run `get file`. The file name remains unchanged. Conversely, you can use `put` to transfer a file from your current directory to a directory of the FTP archive. Of course, this is only possible if you have write permission for the directory. With anonymous FTP, this is usually only the case for a directory with a name like `/pub/incoming`. The FTP session can be terminated using the `quit` or `bye` command. For a reference of the main FTP commands, see Table 11.1.

Command	Function
`?`	Displays a list of all FTP commands
`cd dir`	Changes to the specified FTP directory
`close`	Terminates the connection to the FTP server

Table 11.1 Important FTP Commands

Command	Function
`get file`	Transfers the file from the FTP archive to the current directory
`lcd dir`	Changes the current directory on the local computer
`ls`	Displays the list of files on the FTP server
`lls`	Displays the list of files on the local computer
`mget *.pattern`	Transfers all matching files from the FTP archive to the current directory
`open`	Establishes the connection to the foreign computer (if it did not work at the first attempt)
`put file`	Transfers the file to the FTP archive (*upload*)
`quit`	Terminates FTP
`reget file`	Resumes the transfer of a file that has already been partially transferred

Table 11.1 Important FTP Commands (Cont.)

Other FTP programs

The `ftp` command isn't very easy to use. Fortunately, there are countless alternatives. Most web browsers and file managers available on Linux can also be used for FTP downloads. If you want to work in text mode, you can use the interactive `ncftp` command. The `wget`, `curl`, and `lftp` commands help with the automated transfer of files or entire directory trees via FTP.

11.3.1 SFTP (Secure FTP)

The `sftp` command is part of the `openssh` package. Internally, `sftp` uses a completely different protocol than `ftp` and, like `ssh`, can only be used if an SSH server is running on the remote site. Anonymous FTP is not possible with `sftp`. Apart from that, the operation of the program is the same as that of `ftp`. `sftp -b batchfile` allows you to automate SFTP downloads.

You can also use the Dolphin (KDE) or Nautilus (GNOME) file managers to initiate an SFTP connection by entering the address "sftp://user@servername". After the password request, the programs display the FTP directory as a local directory. Both file managers also directly support the SSH protocol, which works even if `sftp` is not available. For this purpose, you need to specify the address in the following format: `fish://user@servername`.

11.3.2 wget

The interactive approach of the `ftp` command is unsuitable for automating downloads—such as in a script. But `ftp` is also very inflexible in other respects. For example, it's impossible to resume an interrupted download on your own. This can be avoided by using the `wget` command, which is specifically designed to carry out large downloads or transfer entire directories. `wget` supports the FTP, HTTP, and HTTPS protocols in equal measure.

Examples

In its basic form, `wget` simply downloads the specified file:

```
user$ wget ftp://myftpserver.com/name.abc
```

If the download is interrupted for any reason, it can be resumed without any trouble by using `-c`:

```
user$ wget -c ftp://myftpserver.com/name.abc
```

Downloads of large files, like ISO images of Linux distributions, can take several hours if the internet connection isn't very good. That's why it's a good idea to perform the download overnight. The following command almost ensures that the file is actually on the computer the next morning:

```
user$ wget -t 20 --retry-connrefused \
  http://mydownloadserver.com/name.iso
```

Due to `-t 20`, the download is retried up to 20 times after a connection failure. `--retry-connrefused` causes a new attempt to be started even after the *connection refused* error. This is useful if the download server is known to be unreliable and is repeatedly unavailable for short periods of time.

The following command downloads all the files necessary to later read the specified web page offline in an unchanged state:

```
user$ wget -p -k -E -H http://mywebsite.com/page.html
```

Let me briefly explain the meaning of the options: `-p` also downloads CSS files and images. `-k` changes the links in the downloaded files so that they refer to local files. `-E` adds the `.html` identifier to downloaded script files (ASP, PHP, etc.). `-H` also tracks links to external websites.

If you want to read an entire website offline, the following recursive download command (`-r` option) will help. The recursion depth is limited to four levels by `-l 4`:

```
user$ wget -r -l 4 -p -E -k http://mywebsite.com
```

11.3.3 curl

The `curl` command helps to transfer files from or to FTP, HTTP, or other servers. The `man` page lists an impressive range of protocols that `curl` can handle. In this section, however, I will restrict myself to FTP uploads. For script programming, it's particularly useful that `curl` can also process data from standard input or write to standard output. So you don't need to create a **.tar.gz* file first and then transfer it to the FTP server, but can perform both operations simultaneously using a pipe.

The following command transfers the specified file to the FTP server `backupserver` and saves it in the `dir` directory:

```
user$ curl -T file -u username:password ftp://backupserver/dir
```

To process data from the standard input channel, you can use `-T` to specify a hyphen as the file name. The following command saves the result of the `tar` command directly in the `name.tgz` file on the FTP server:

```
user$ tar czf - dir/ | curl -T - -u usern:pw ftp://bserver/name.tgz
```

`curl` is also well suited for testing REST APIs. In the following example, a service on the *https://ipinfo.io* website tries to determine your location based on the IP address of your call:

```
user$ curl https://ipinfo.io
  {
    "ip": "91.115.157.28",
    "hostname": "91-115-157-28.adsl.highway.telekom.at",
    "city": "Graz",
    "region": "Styria",
    "country": "AT",
    "loc": "47.0667,15.4500",
    "postal": "8041",
    "timezone": "Europe/Vienna", ...
  }
```

11.3.4 lftp

`lftp` is a convenient interactive FTP client. However, the command is also well-suited to execute FTP uploads or other commands in a script. For this purpose, you can either use `-c` to pass several FTP commands separated by semicolons to `lftp` or use `-f` to specify a file that contains these commands line by line. The first command will always be `user username,password`

servername to connect to the FTP server. The following command demonstrates a file upload:

```
root# lftp -c "open -u username,password backupserver; put www.tgz"
```

If you want to give the file a different name on the FTP server, you need to additionally specify the -o <newName> option. lftp displays the current progress during the upload.

To transfer an entire directory to the backup server instead of a file, you can use the mirror -R command. This command usually copies directories from the FTP server to the local computer. -R reverses the transfer direction. Here is another example:

```
root# lftp -c "open -u usern,passw bserver; mirror -R directory"
```

Unlike other FTP clients, lftp supports the du command, which you can use to determine how much disk space your backup files already consume. This is important if your storage space on the backup server is strictly limited. The following command shows how you can determine the amount of memory already used without interactive intervention. The -s option indicates that you are only interested in the final sum. -m ensures that MiB is used as the unit of measurement:

```
user$ lftp -c "open -u username,password bserver; du -s -m"
2378    .
```

If you want to use the result for a calculation, the second column (i.e., the dot indicating that the numerical value refers to the current directory) will interfere. Simply add cut -f 1 after the command to extract the first column:

```
user$ lftp -c "open -u usern,passw bserver; du -s -m" | cut -f 1
2378
```

11.3.5 rsync, mirror, and sitecopy

rsync helps you to copy or synchronize entire directory trees. For a detailed description of this command, see Chapter 28, Section 28.6. If neither an SSH nor an rsync server is running on the partner computer, you can use the mirror or sitecopy commands instead of rsync. The Perl script mirror from the package of the same name copies entire directory trees from an FTP server to the local computer. The sitecopy command, on the other hand, is optimized to upload a directory tree to a web server, with the data transfer taking place either via FTP or WebDAV.

11.4 Lynx

Web browser in text mode

Web browsers such as Firefox or Chrome are unusable in a text console or terminal window. Nevertheless, to quickly visit a web page or read an HTML document even in text mode, programs such as ELinks, Lynx, or w3m can help you. You also can use these programs to convert simple HTML documents into plain text. All three programs are similar in operation. Numerous options as well as keyboard shortcuts are documented in the `man` pages or in the integrated help system.

Lynx

The operation of Lynx is simple: You usually start the program by specifying a WWW address or the name of an HTML file as a parameter. Lynx loads the document and displays the first page, with headings and links indicated by different colors. If you start Lynx with the `-use_mouse` option, you can also operate the program by mouse: you can follow a link with the left button, the middle button displays a context menu, and the right button takes you back to the previous page. Lynx uses the Latin-1 character set for output by default. To ensure that special characters are displayed correctly in Unicode consoles, you must specify the `-display_charset=utf-8` option. The following command shows how you can use Lynx as a converter from HTML to plain text:

```
user$ lynx -dump source.html > target.txt
```

11.5 Mutt

The text-based email program Mutt can be used to read local emails (see Figure 11.2). The package, usually of the same name, must be installed prior to its first use. In a console window, you first need to run `su -l` to log in as `root`, and then start the program using the `mutt` command.

The program displays the subject lines of all emails on the start page. If the active user hasn't received a single email yet, Mutt complains that the `/var/mail/user` file does not exist yet. You can ignore this warning; it will stop appearing as soon as the first email arrives.

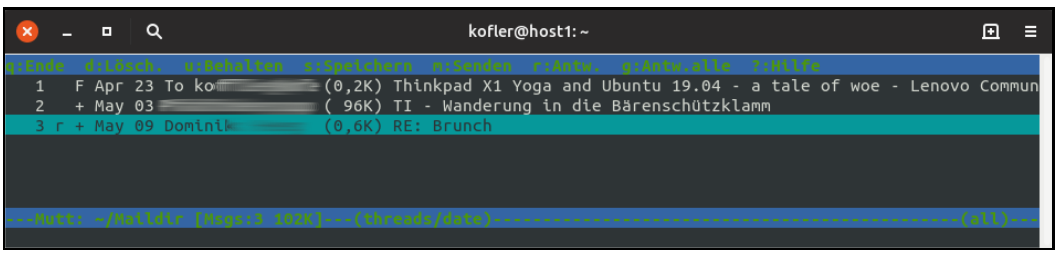


Figure 11.2 Reading Local Emails in Mutt

You can use the cursor keys to move through the inbox. Enter displays the text of the selected email. The space bar allows you to scroll through the message. J takes you to the next message, I back to the inbox, and ? displays a help text with all important keyboard shortcuts.

To compose a new email, you want to press M and specify the recipient and the subject line. Mutt then starts the editor that you selected using the `$EDITOR` environment variable or via the `/etc/alternatives/editor` link. There you write the message text, save it, and exit the editor. Then send the email in Mutt by using Y. Q terminates the program. Upon terminating, Mutt asks two questions: Should emails that have been marked as deleted with D be permanently deleted? And should messages that have been read be moved to `/home/username/mbox`? If you plan to edit the emails later in a different program, you should answer N to both questions. The second question in particular is critical: in the local `mbox` file, only Mutt finds the emails, not an external program such as the Dovecot POP server.

Configuration

Mutt works right out of the box if your email is in an mbox file in the `/var/mail/name` directory. On the other hand, if your emails are stored in Maildir format in the `Maildir` directory, you need to set up the `/etc/Muttrc` or `.muttrc` configuration file with the following content:

```
# File /etc/Muttrc (global) or .muttrc (user-specific)
set mbox_type=Maildir
set folder="~/Maildir"
set mask="!^\\.[^.]"
set mbox="~/Maildir"
set record="+.Sent"
set postponed="+.Drafts"
set spoolfile="~/Maildir"
```

More Maildir configuration tips for various special cases can be found at the following websites:

- *https://gitlab.com/muttmua/mutt/wikis/MuttFaq/Maildir*
- *https://eising.wordpress.com/mutt-maildir-mini-howto*

Sender host name (hidden_host)

If you write an email as `peter` in Mutt, the program typically uses `peter@hostname` as the sender address. On Debian and Ubuntu, the contents of the `/etc/mailname` file are taken into account instead of the hostname, if it exists. If a computer is configured as a mail server, then this file contains the domain for which the mail server is responsible.

Sometimes you won't want Mutt to use the full host name. Maybe the host name of your mail server is `host1.a-company.com`, for example, but emails that you compose in Mutt as `peter` should use only `one-company.com` as the

host name. On Debian and Ubuntu, this works automatically thanks to `/etc/mailname` (see Chapter 26, Section 26.2). On other distributions, you get equivalent behavior if you include `set hidden_host=yes` in `/etc/Muttrc` or in `.muttrc`. This way, Mutt uses the domain name instead of the full host name.

If you're unsure which setting applies to a particular Mutt option, you should press [:] and then run the command `set ?optionsname`. A good overview of the most important Mutt settings can be found on the Arch wiki at *https://wiki.archlinux.org/title/mutt*.

PART IV

Text and Code Editors

Chapter 12
Vim

The focus of this chapter is the Vi editor and its open-source implementation, Vim (*Vi Improved*). These editors are relatively difficult to learn, as is the Emacs editor presented in the next chapter. But this effort is worth it if you frequently edit text, program code, configuration files, and the like—that is, if a text editor is a constant and indispensable tool for you.

Vi and Emacs have the advantage that the programs can be executed in text mode in a console if required—that is, also in an SSH session. That is why administrators in particular prefer Vi or Emacs to modern editors that necessarily require a graphical user interface.

That still leaves us with the question of all questions: Vi or Emacs? The programs are primeval rocks of the Unix/Linux history. Both tools provide countless special functions, such as automatic syntax highlighting for numerous programming languages and document types or search and replace with regular expressions. There have been endless discussions on the web about which program is better. I can't really answer the question objectively: because written all other chapters of this book of this book with Emacs variants (except this chapter, of course), I am more familiar with Emacs than any Vi variants.

Personally, I find the Emacs editor more intuitive to use and easier to learn. With Vi, the distinction between standard and insert mode can drive you crazy at first. On the other hand, the fact that the program is a true standard on Unix/Linux speaks for Vi and its variants. It uses significantly fewer resources and is available even on minimal rescue systems. Real Unix/Linux freaks should know both editors in their basic functions anyway—and I don't teach much more than that in this book.

The original Vi editor is a commercial program and therefore not available on Linux. Vim, on the other hand, is an open-source program that is compatible with Vi and also provides countless improvements and extensions. The program can be started with the command `vi` or `vim`.

Vim in graphics mode

Basically, Vim is executed in a text console or console window. If you prefer a proper menu and neat scroll bars, you should take a look at `gvim` (see Figure 12.1). This graphical variant to Vim may need to be installed separately, and the package name is often `vim-gnome` or `vim-gtk3`.

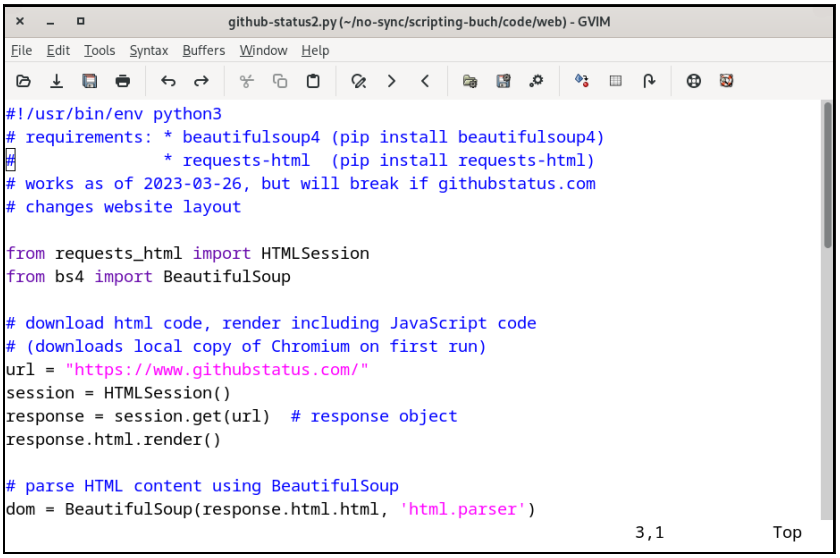


Figure 12.1 Graphical Variant of Vim Editor

Links

Due to space limitations, this chapter can only provide an introduction to Vim. The tutorial, which you can start using the `vimtutor` command, is very helpful for beginners. (This starts Vim and loads a help text containing an introduction and examples to try out). The following links reference pages with further information:

- *https://www.vim.org* (homepage)
- *https://vimhelp.org/vim_faq.txt.html* (FAQ)
- *https://fprintf.net/vimCheatSheet.html* (keyboard shortcuts)
- *https://truth.sk/vim/vimbook-OPL.pdf* (500-page Vim book)

Vim is charityware

Lead developer Bram Moolenaar refers to Vim as *charityware*. Vim is available free of charge under a GPL-compatible license. However, those who use Vim regularly are asked to show their gratitude in the form of a donation, which will go to a children's charity in Uganda. Further information is provided by the Vim command Esc : help iccf Enter.

12.1 Quick Start

You usually start Vim by entering "vim [file name]" within a text console or in a console window. The file to be modified is displayed directly in the console.

Standard mode

Before you can start writing, however, you need to deal with a peculiarity: the program distinguishes between different editing modes (see Table 12.1).

The default mode is not for entering text, but for executing commands. For example, if you type L in standard mode, this will move the cursor one character to the right. D, W deletes a word; P reinserts it at the current cursor position; and so on.

Insert mode

To enter text, you need to switch to the insert mode via I (*insert*) or A (*append*). vim displays the text -- INSERT -- in the left-hand corner of the bottom line. In insert mode, you can enter text, move the cursor, and delete individual characters (Del and Backspace). The difference between I and A is that with I the input starts at the current cursor position, and with A at the character behind it.

Before you can enter a command again, you must press Esc to return to standard mode. This mode is not marked separately, so the left-hand part of the bottom line is then left blank.

Changing Modes without Changing the Cursor Position

When switching from insert mode to standard mode, the cursor moves one character to the left—unless it is already at the beginning of a line. According to the Vim FAQs, this strange behavior is intentional and cannot be prevented. To execute a single command without leaving the insert mode—and thus without changing the current cursor position—you need to initiate the command input using Ctrl+O.

Deleting text

In insert mode, you can use Del and Backspace to delete individual characters as usual. If you want to delete words, lines, or entire areas, you must first switch to standard mode by pressing Esc. Then D, W deletes a word and D, D deletes an entire line. If you prefix a number, the delete command is repeated a corresponding number of times. 5, D, D thus deletes five lines. . repeats the command that was executed most recently. P (*put*) inserts the most recently deleted text after the current cursor position, Shift+P before it. U (*undo*) undoes the last changes, Ctrl+R (*redo*) restores the changes.

Keyboard Shortcut	Function
I	Activates the insert mode.
A	Activates the insert mode. The text input starts at the next character.
Esc	Activates the standard mode or cancels the command input.

Table 12.1 Basic Commands

Keyboard Shortcut	Function
Commands in Standard Mode	
D, W	Deletes a word.
D, D	Deletes the current line.
n D, D	Deletes *n* lines.
P	Inserts the last deleted text after the cursor position.
Shift+P	Inserts the most recently deleted text before the cursor position.
.	Repeats the last command.
U	Undoes the last change.
Shift+U	Undoes all changes in the current line.
Ctrl+R	Undoes the undo (*redo*, as of Vim 7).
: w	Saves the file.
: q	Exits Vim.
: q!	Exits Vim even if there are unsaved files.
Commands in Insert Mode	
Ctrl+O *command*	Executes the command without leaving the insert mode.

Table 12.1 Basic Commands (Cont.)

Saving and exiting

To save the changed file, you need to switch to standard mode by pressing Esc and then enter the following command: : w Enter (*write*). : q Enter (*quit*) exits the editor if all open files are saved. Using : q! Enter, you can force an end even if there are unsaved changes. : wq Enter combines saving and exiting the program.

12.1.1 Help

Vim provides comprehensive online help. You can get to the start page of the help system from any mode by pressing F1. Alternatively, in standard mode, use : help or : help *topic* to open the help. If you want to know what help topics there are that contain the keyword *abc*, for example, you need to type : help *abc* Ctrl+D.

The help window

The help text is displayed in a separate section of Vim (a so-called window, even though it isn't a separate window in the sense of the Linux graphics

system). You can close this window again using : q. However, you can also leave the help window open and continue working in the original text. To do this, switch the currently active window using Ctrl+W, W. More information on handling Vim windows and buffers and editing multiple files follows in Section 12.5.

Navigating the help

In the help text, references to other help topics are highlighted (light blue in the version I tested). To jump to this topic, you must move the cursor to the keyword and run Ctrl+]. It is even easier if the mouse is activated (see Section 12.7): in that case, double-clicking the help topic is enough to go there. Ctrl+T returns you to the original page.

12.2 Cursor Movement

The cursor keys work in standard mode as well as in insert mode. You can also change the cursor position using various key combinations in standard mode (see Table 12.2). Vi freaks move through the text more efficiently in this way than by using the cursor keys.

A peculiarity of Vim is that ← at the beginning of a line doesn't place the cursor at the end of the previous line. Similarly, → at the end of a line doesn't work as usual. To achieve the usual behavior of other editors, you need to execute in default mode, with : `set whichwrap=b,s,<,>,[,]`, or add this `set` command to `.vimrc`.

Saving cursor positions

M <letter> stores the current cursor position in a position marker. Press ' <letter> to move the cursor back to the position thus saved.

Vim remembers the cursor position where a new cursor movement starts. ', ' leads back to this position. Once again ', ' moves the cursor back to the last position. ', [or ',] move the cursor to the beginning or end of the last changed text section.

Where am I?

In insert mode, Vim displays the current row and column numbers in the status bar on the right, as well as a percentage indicating which section of the text you are in (e.g., 92%). Via Ctrl+G, Vim also shows in the status line the name of the file, its state (such as *modified*), the total length in lines, and the relative position in the text in percent.

Keyboard Shortcut	Function
Cursor keys	The cursor keys have the commonly known meaning.
H/L/J/K	Moves the cursor left/right/down/up.

Table 12.2 Keyboard Shortcuts for Cursor Movements in Standard Mode

Keyboard Shortcut	Function
Shift+H/Shift+L	Moves the cursor to the beginning or end of the current page.
Shift+M	Moves the cursor to the center of the current page.
B/W	Moves the cursor one word to the left/right.
E	Moves the cursor to the end of the word.
G, E	Moves the cursor to the beginning of the word.
[,]	Moves the cursor to the beginning of the current/next block.
{, }	Moves the cursor to the beginning of the current/next paragraph.
^, $	Moves the cursor to the beginning or end of the line.
Shift+G	Moves the cursor to the end of the line.
G, G	Moves the cursor to the beginning of the line.
n Shift+G	Moves the cursor to line *n*.
n	Moves the cursor to column *n*.
%	Moves the cursor to the corresponding bracket: ()[]{}.

Table 12.2 Keyboard Shortcuts for Cursor Movements in Standard Mode (Cont.)

12.3 Editing Text

Inserting text characters multiple times

To insert a text character more than once, you must enter the number in standard mode, the command A (*append*), the relevant character, and finally press Esc. So to enter the = character 50 times, you need to enter 5, 0, A, =, Esc. Once you have entered the command, you will be back in standard mode.

Typos

Vim also helps you to correct typical typos: ~ changes the case of the current letter, and X, P swaps the following two letters.

Deleting text

Table 12.3 provides an overview of the most important commands for deleting text. If you enter a number before the delete command, the delete command will be repeated a corresponding number of times. As for all other Vim commands, . repeats the last command, and *n* . repeats it *n* times.

Keyboard Shortcut	Function in Insert Mode
Del, Backspace	These keys have the usual meaning.
Function in Standard Mode	
X	Deletes the character at the cursor position or the selected text.
Shift+X	Deletes the character before the cursor.
D, D	Deletes the current line.
D *cursorcommand*	Deletes the text according to the cursor movement command (see Table 12.2). *Examples:* D, $ deletes everything to the end of the line. D, B deletes the previous word. D, W deletes the next word.

Table 12.3 Deleting Text

Text is always deleted into a copy register. From there, the most recently deleted text can be reinserted into the text by using Shift+P at the current cursor position or using P behind the cursor position.

A peculiar way to delete text and then replace it with new text is provided by the C commands (*change*): For example, C, W deletes the current word and activates the insert mode. You then enter the new word and finish the entry by pressing Esc. C works similarly for other cursor commands.

Copying text

You can also insert text into the copy register without deleting it. Table 12.4 summarizes the corresponding commands. All commands apply to the standard mode.

Selecting text

Some (delete) commands require you to first select a text section. Vim provides three different selection modes, which you can activate or deactivate using V, Shift+V or Ctrl+V at the starting point of the selection. While one of these modes is active, the bottom Vim line contains the text `-- VISUAL --`.

You then move the cursor to the endpoint of the selection or expand the selection using some specific selection commands (see Table 12.5).

As long as the selection mode is active, various commands are available for editing the selected text (see Table 12.6).

Keyboard Shortcut	Function
Y	Copies the selected text to the copy register.

Table 12.4 Copying Text to the Copy Register

Keyboard Shortcut	Function
Y, Y	Copies the current line to the copy register.
Y *cursorcommand*	Copies the text captured by the cursor movement. *Example:* Y, } copies the text to the end of the paragraph.

Table 12.4 Copying Text to the Copy Register (Cont.)

Keyboard Shortcut	Function
V	(De)activates the character selection mode.
Shift+V	(De)activates the line selection mode.
Ctrl+V	(De)activates the block selection mode.
A, W	Extends the selection by one word.
A, S	Extends the selection by one block.
A, P	Extends the selection by one paragraph.
A, B	Extends the selection by one () level.
A, Shift+B	Extends the selection by one {} level.
G, V	Selects the most recently selected text again.
O	Toggles the cursor position between the start and end of the selection.

Table 12.5 Selecting Text

Keyboard Shortcut	Function
X	Deletes the selected text.
Y	Copies the selected text to the copy register.
~	Changes uppercase/lowercase.
Y	Merges the selected lines into one long line.
G, Q	Performs a line break (for continuous text).
>, <	Indents or outdents the text by one tab position.
=	Indents the text according to the current `indent` mode.
`!sort`	Sorts the lines using the external `sort` command.

Table 12.6 Editing Selected Text

Indenting lines

Especially when editing code, the correct indentation of lines is important.

Vim helps you with this in many ways. The most basic commands are [>], [>] or [<], [<]. You move the current line in or out by one tab position. If you select several lines of text beforehand, you can apply the commands to an entire block. The simple input of [>] or [<] is sufficient. [:] `set shiftwidth=N` changes the indentation depth (typically eight characters).

Vim can also attempt to automatically indent new lines as they are typed. To do this, you need to activate an indent mode—for example, by [:] `set cindent`. The following is a brief summary of the basic functions of the main Vim indent modes:

- **autoindent**
  Indents the subsequent line as far as the previous one.
- **smartindent**
  Works like `autoindent`, but additionally considers {} bracket levels. For Vim to recognize the closing brackets correctly, they should be specified at the beginning of a new line. You can control the extent of indentation depending on the bracket level using the `shiftwidth` option. Previously selected text can be reindented by [=].
- **cindent**
  Works like `smartindent`, but also takes into account various code structures of C or C++. The indenting mechanism can be adjusted to personal preferences via various options (see [:] `help C-indenting`).

Continuous text

Vim is preconfigured so that writing code or changing configuration files works as well as possible. For this reason, Vim doesn't perform an automatic line break (i.e., you must start new lines yourself by pressing [Enter]). However, you can of course use Vim to compose ordinary text (such as for emails). Table 12.7 summarizes some special commands and options that can help with this.

Keyboard Shortcut	Function
[Shift]+[J]	Connects the current line with the subsequent one
n [Shift]+[J]	Connects *n* lines to one long line
[G], [Q], [A], [P]	Breaks the current paragraph again and places the cursor at the beginning of the next paragraph
[G], [W], [A], [P]	As above, but leaves the cursor at the current position

Table 12.7 Editing Continuous Text

Keyboard Shortcut	Function
: set textwidth=nn	Automatic line break after a maximum of *nn* characters (normally: 0 = disabled)

Table 12.7 Editing Continuous Text (Cont.)

The G commands take into account the autoindent mode as well as the textwidth setting. If textwidth contains 0, the maximum line length is 79 characters. Configuration options for particularly convenient continuous text input are provided by the formatoptions option (see the associated help text).

Word complements

Typing long words and function and variable names is tedious and prone to errors. Vim helps you do this in an ingenious way: you just need to type the first letters and then press Ctrl+P.

If the word is already clearly identified, it will be completed immediately. Otherwise you can select the required word using Ctrl+P or the cursor keys. Vim takes all words of all loaded files into account for the word completion function, preferring words from the current file and especially words near the cursor position.

12.4 Search and Replace

Searching text

In the default mode, / *searchtext* Enter moves the cursor to the text being searched for. N repeats the search, while Shift+N repeats the search backwards. To search backwards from the beginning, you want to start the search using ? *searchterm*. Table 12.8 explains the most important special characters you can use to search for patterns.

Character	Meaning
.	Any character.
^ $	Start of line/end of line.
\< \>	Start of word/end of word.
[a-e]	A character between *a* and *e*.
\s, \t	A space or a tab character.
\(\)	Summarizes a search pattern as a group.
\=	The search term must occur 0 times or once.

Table 12.8 Special Characters in Search Term

Character	Meaning
*	The search term may occur any number of times (even 0 times).
\+	The search term must occur at least once.

Table 12.8 Special Characters in Search Term (Cont.)

Case sensitivity

Vim is case-sensitive by default when searching. If you don't want this, you must initiate the search pattern with `/c` (applies only to this search) or run `:` `set ignorecase` (applies to all subsequent searches).

Incremental search

You can use `:` `set incsearch` to activate the incremental search: Vim moves the cursor to the first matching location as soon as you enter the search text using `/` *searchterm*. `Enter` ends the search; `Esc` cancels it. After the search, all matches in the text remain highlighted until you perform a new search or run `:` `nohlsearch`.

Search and replace

To replace all occurrences of the text *abc* with *efg* without prompting, you need to run `:` `%s/abc/efg/g` in standard mode. `'`, `'` then returns to the beginning of the search. Table 12.9 presents some variants of the search-and-replace command. When searching and replacing with a query, you can use `Y` or `N` for each search term found to specify whether it should be replaced by the new text or not. `Q` cancels the operation; `A` replaces all further occurrences. In the replace term, you can use *n* to refer to the *n*th group in the search pattern. You can find a lot more tips on searching and replacing along with numerous examples in the online help (`:` `help substitute`).

Keyboard Shortcut	Function
`:` `%s/abc/efg/g`	Replaces all occurrences of *abc* by *efg* without a query
`:` `%s/abc/efg/gc`	Replaces all occurrences of *abc* with *efg* with a query
`:` `%s/abc/efg/gci`	Replaces without considering uppercase or lowercase

Table 12.9 Search and Replace

12.5 Editing Multiple Files Simultaneously

In default mode, `:` `e filename` loads a new file. The new file replaces the currently edited file, which you must save prior to that, as otherwise Vim will abort the operation. You can force loading the new file using `:` `e! filename`, but then you will lose all changes made to the last current file.

Buffers and windows

Of course, you can also edit multiple files at the same time in Vim. Before that, however, you should understand the not particularly intuitive concept of how Vim internally manages and displays text. Each text displayed in the editor is internally located in a buffer.

This applies to both files and help texts. So long as there is only one buffer, it will be displayed in the entire Vim workspace. To display multiple buffers at the same time, the workspace is divided into several windows, as is the case when you display help texts. Such a window is not an independent window in the sense of the Linux graphics system, but only a subarea of the workspace.

The buffers of modified files are always visible in a window. On the other hand, buffers of files that haven't been changed since the last save can be hidden. The buffers remain in the memory but are no longer considered active. (*Use caution!* If you close a window with a file that hasn't yet been saved, all changes will be lost! The file's inactive buffer remains available but contains the file as it was at the time of the last save.) Vim is also able to display a file in multiple windows at the same time—for example, to edit different parts of a very long text.

Tabbed window

Vim facilitates the editing of multiple files by using dialog sheets in text mode (see Figure 12.2). This works really conveniently if you've activated the mouse (see Section 12.7). Then you can easily select the currently active file using the mouse or close it via the **X** button in the top-right-hand corner. For more information about these *tabbed pages*, use `:` `help tabpage`.

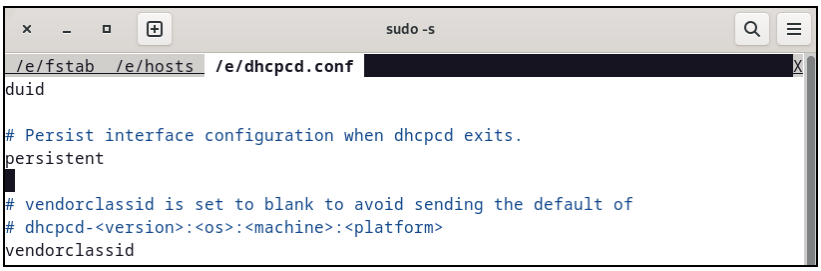


Figure 12.2 Three Files in Three Tabbed Windows

Loading a new file

Depending on whether you want to work with windows or tabbed windows, you can load new files using `:` `new filename` or `:` `tabnew filename`. If you pass multiple files at program startup—for example, `vim file1 file2 file3`—only the first file will be displayed; the other files will be loaded into invisible buffers.

To open each file in a window or tabbed window, you must also pass the `-o` or `-p` option. Table 12.10 summarizes the most important commands for loading and saving files, switching between (tabbed) windows, and so on.

Keyboard Shortcut	Function
: e filename	Loads a file into the current buffer
: w	Saves the current file
: wall	Saves all open files
: wq	Saves and closes the buffer
: q	Closes the current buffer and exits Vim when no more buffers are open
: q!	Closes the buffer even with unsaved changes
: qall	Closes all buffers and exits Vim
: split	Splits the window and displays the same text in both parts
: new	Creates an empty buffer and displays it in a window
: new file	Loads a file into a new buffer
: only	Maximizes the current window and closes the other buffers
: all	Shows all buffers in correspondingly reduced windows
: buffers	Returns the list of all buffers
: buffer n	Displays buffer *n* and deletes the current buffer
: buffer file	Displays the buffer with the file in the current window
: tabnew	Creates a buffer and displays it in a tabbed window
: tabnew file	Loads a file and displays it in a tabbed window
: tabnext	Switches to the next tabbed window
: tabprevious	Switches to the previous tabbed window
: tabclose	Closes the current tabbed window
: tabonly	Closes all other tabbed windows

Table 12.10 Files, Buffers, and Windows

12.6 Internal Details

In the previous sections, special functions of Vim were repeatedly activated or modified by options, and this section introduces a number of additional options.

Options

Note that some options apply only locally to the current file or buffer. In this case, : set only changes the local setting. To change the option globally, you need to use : setglobal. Table 12.11 summarizes the most important commands for editing options.

Keyboard Shortcut	Function
: help options	Provides general information as well as an option reference
: help 'option'	Provides information about the specified option
: set	Returns all options that are not in the default state
: set <*booleanoption*>	Enables the Boolean option
: set no<*booleanoption*>	Disables the Boolean option
: set inv<*booleanoption*>	Inverts the current state of the option
: set option?	Shows the setting of the option
: set option=value	Resets the option
: set option+=value	Changes the option
: set option-=value	Changes the option
: set option&	Resets the option to the default state

Table 12.11 Working with Options

Configuration (vimrc)

All option settings are lost when you exit Vim. To set options permanently, you need to modify the Vim configuration file, .vimrc. Note that comments must be introduced by the " character! The following lines contain a simple example of what .vimrc can look like. All options used there have been or will be presented in this chapter. If you search the internet for vimrc, you will find countless other examples:

```
" Example for ~/.vimrc
" Set cursor position with the mouse
set mouse=a

" <cursor left> at the beginning of the line moves the cursor to the
end of the previous line,
" <cursor right> at the end of the line moves the cursor to the
beginning of the next line
```

```
set whichwrap=b,s,<,>,[,]
" Create backups (name~) on save
set backup
" activate incremental search
set incsearch
" generally insert spaces instead of tabs
set expandtab
```

The configurability of Vim goes much further still. You can set Vim options differently for different file types, program new functions yourself, and so on. But don't try to reinvent the wheel! At *https://www.vim.org/scripts*, you'll find countless ready-made solutions that you can use with little effort. It's usually sufficient to include the relevant file in `.vimrc` by using `source filename`.

Regardless of the backup setting, Vim periodically updates a swap file while you are writing. The name of this file is a combination of a preceding dot, the current file name, and the `.swp` extension (e.g., `.mycode.c.swp` if you are currently editing `mycode.c`). This file contains all the changes you have made since the last save in a binary format. It's automatically updated when you've entered more than 200 characters of new text or haven't made any entries for four seconds. The swap file is deleted when Vim is exited in a regular way.

If the power fails, Linux or Vim crashes, or some other problem strikes, you can restore your unsaved work the next time you start Vim. Vim notices that a swap file exists when the file is opened and provides various options to choose from. Typically, you will choose W (restore) and then save the file thus restored. After that, you should exit Vim and explicitly delete the swap file. (This doesn't happen automatically!)

Character set

Vim handles most major character sets by default, including various Latin variants, Unicode (`utf-8`, `utf-16`, `ucs-2`, `ucs-2l`, `ucs-4`, etc.), and some Asian two-byte character sets (such as `euc-kr`—that is, Korean).

Vim determines the default character set of the operating system during the startup process (`encoding` option). When reading a new file, Vim tries to detect its character set as well (`fileencoding` option). All character sets listed in the `fileencodings` option are tried in turn until a character set is found for an error-free display of the entire text. (The default setting for `fileencodings` is often `utf-8,latin1`.) To determine what character set the file you are currently editing uses, you want to run : `set fileencoding?`. You can use : `set fileencoding=<newcharacterset>` to change the character set. If you then save the file, it will be saved in the new character set!

12.7 Tips and Tricks

Entering : commands efficiently

When entering commands that start with [:], there are some entry helpers: You can use the cursor keys to scroll through the most recently used commands. (Vim stores the commands in `.viminfo` and thus remembers the commands even after the program ends.) Furthermore, you can complete keywords using [Tab] (e.g., when entering options). And finally, you can abbreviate many [:] commands (such as [:] `tabn` instead of [:] `tabnext`).

Displaying line numbers

[:] `set number` displays the line number next to each line, while [:] `set nonumber` deactivates this mode again.

Backups

By default, Vim doesn't create a backup (i.e., a copy of the original file) upon saving. If you wish to do so, you need to run [:] `set backup` in standard mode. The backup file is given the name `alternname~`. To enable backups in general, you can insert `set backup` in `.vimrc`.

Activating the mouse

If you use Vim in a text console or in a console window, then the function of the mouse is limited to its basic functions on X: you can use it to copy and paste text at the current cursor position, but you cannot change the current cursor position, and so on.

To provide the mouse with more functions in Vim, you can either use the graphical `gvim` variant or run [:] `set mouse=a` in standard mode. Then you can reposition the cursor via a mouse click, select the active Vim window, scroll through the text using the mouse wheel, and so on.

However, this mouse mode has a disadvantage: the middle mouse button then inserts the most recently deleted text in Vim. The mouse can no longer be used to copy text between Vim and other programs. But you can easily solve this: the conventional mouse functions are still available if you also press the [Shift] key. But make sure that Vim is actually in insert mode when you insert text! Otherwise, the text inserted using the mouse will be interpreted as a command, and that can go wrong.

Spaces instead of tabs

To make Vim always insert spaces instead of tab characters in your text, you should run [:] `set expandtab` or add the statement to `.vimrc`. To replace all tab characters in the existing file with the appropriate number of spaces, you then need to run [:] `retab`. Conversely, to replace spaces with tabs, you must run [:] `set unexpandtab` and then [:] `retab!`.

Macros

[.] repeats the last command; you already know that. But if you want to execute a whole sequence of commands multiple times, you should define a macro. To do this, start the macro mode in standard mode via [Q].

The next character specifies the name of the macro (to be precise, the name of the register where the macro is stored). All other commands are stored in the macro until you end the input by pressing [Q] again. After that, you can

run the macro recorded in this way by using @ *macroname*. (Once you exit Vim, all saved macros will be lost.) For example, the following key sequence records the macro *a*, which inserts the quotation mark " at the beginning and end of a word: Q, A, I, ", Esc, E, A, ", Esc, Q. If the cursor is now inside a word and you execute @, A, then this word will be put in quotes. @, @ repeats the last register command without you having to remember the macro or register name.

Running Linux commands

If you want to execute a Linux command in Vim without leaving Vim, you must run : `!commandname` in standard mode (e.g., `!ls` to get the list of files in the current directory). Vim then displays the result of the command. You can press Enter to return to the editor. To execute multiple commands, you must open a new shell using : `sh`. From there, Ctrl+D will take you back to the editor.

Using Vim in easy mode

If you can't get used to the different Vim modes, but don't want to do without Vim anymore, it's best to start the program via `vim -y` or run : `set insertmode`. This way the editor always will remain in insert mode. This means that you must initiate each command with Ctrl+O. To execute multiple commands at once, you can also switch to standard mode for a longer period of time by pressing Ctrl+L and exit it by pressing Esc.

Vim behaves even more like other editors when you launch `evim`: this will start the editor in the graphical user interface, `gvim`. Text selections can be made using Shift and the cursor keys. Texts can be copied using Ctrl+C, deleted using Ctrl+X, and pasted using Ctrl+V. `man evim` refers to the easy mode as “Vim for gumbies” (whatever a *gumby* is). You'll have to make do without so many of the basic features of Vim that it makes more sense to use a different editor right away.

Chapter 13
Emacs

To call Emacs simply an editor is not enough: the program is suitable not only for editing texts, but also as a complete development environment and, die hard Linux fans, even as an email client. Those who have grown up with Unix use Emacs as a substitute operating system, as it were, for all the functions of everyday work. In my case, it isn't quite so dramatic, but the fact is that I have written this book (and many others) in Emacs since the first edition!

Where there is a lot of light, there are also shadows. The operation of Emacs looks daunting at first glance: it's teeming with Ctrl and Alt sequences that are used to call the countless commands. It's not without reason that mockers claim the name Emacs stands for *Escape Meta Alt Control Shift*. (As a matter of fact, Emacs stands for *Editor MACroS*.) At the same time, the user interface (menu, toolbar) looks antiquated. The configuration is cumbersome. Long story short: Emacs is an editor for professionals who are willing to invest time in learning it and who are not bothered by appearances.

13.1 Quick Start

This chapter focuses on GNU Emacs (see Figure 13.1). **Variants**

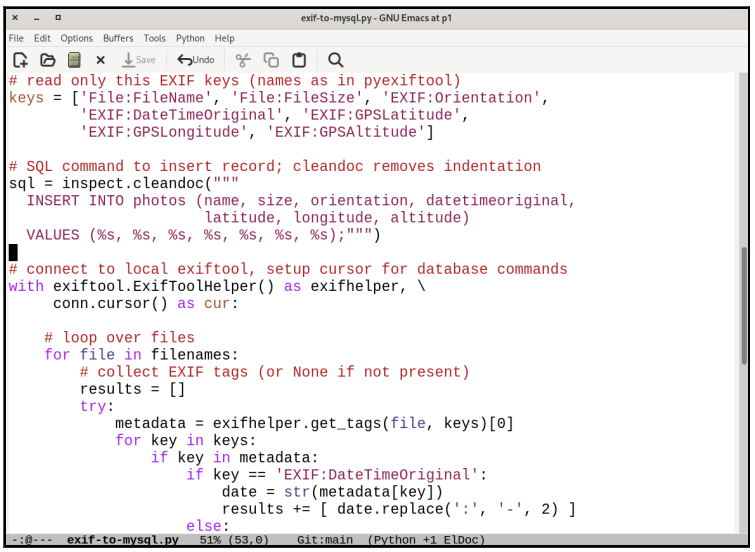


Figure 13.1 GNU Emacs

The Emacs fork XEmacs, which had a wider distribution for a while, no longer plays a role in everyday Linux. Much more interesting are the "small" Emacs variants like `jed`, `jmacs` (from the `joe` package), `jove`, or `zile`: their main advantage is that their resource requirements are much lower. This makes these programs ideal for emergency systems or on servers where a configuration file only needs to be changed now and then.

Links

For more information on Emacs, please visit the following websites:

- *https://www.gnu.org/software/emacs/emacs.html*
- *https://www.emacswiki.org*

13.1.1 Loading and Saving Texts and Exiting the Program

Loading and saving files

In Emacs, you can load a file using Ctrl+X, Ctrl+F *filename* Enter. Ctrl+X, Ctrl+S saves a modified file. To save a file under a different name, you need to type Ctrl+X, Ctrl+W *filename* Enter.

Keyboard Shortcut	Function
Ctrl+X, Ctrl+F *file* Enter	Loads a file (find)
Ctrl+X, I	Inserts a file into existing text (insert)
Ctrl+X, Ctrl+S	Saves a file (save)
Ctrl+X, S	Saves all files (with query)
Ctrl+X, S, !	Saves all open files (without query)
Ctrl+X, Ctrl+W *file* Enter	Saves under a new name (write)
Ctrl+X, Ctrl+C	Exits the editor

Table 13.1 Loading and Saving Files; Exiting Emacs

Undo

You can use Ctrl+X, U (*undo*) or Ctrl+^ to undo the most recent changes. This undo function works for any complex commands and is practically unlimited!

If you make a mistake while entering a command, you can cancel the command entry by using Ctrl+G. This is especially useful if you press Esc by mistake.

End of program

Ctrl+X, Ctrl+C terminates Emacs. If there are any unsaved files, a confirmation prompt will appear asking if you really want to exit Emacs without saving. Respond to this prompt by typing "yes" and pressing Enter if you actually want to discard the changes. If you use Emacs in a

text console, you can temporarily exit the program by pressing Ctrl+Z. fg then lets you resume the work. In graphics mode, Ctrl+Z only causes the image to be reduced to an icon.

Backups

Emacs automatically creates a backup *name*~ containing the original text when saving. Also, Emacs periodically saves the current state of the text in the #name# file. You can use this file if a power failure occurs while you're working or if you can't exit Emacs properly for some other reason.

13.1.2 Online Help

Emacs provides numerous commands for calling the online help. The most important command for getting started is F1, T (*tutorial*). Ctrl+X, B, Enter will return you to the original text.

If after executing a help command, multiple text sections (windows) remain, you can use Ctrl+X, O (the letter O) to place the text cursor in the next window in each case. Ctrl+X, 0 (zero) removes the current window; Ctrl+X, 1 deletes all windows except the current window. So with these three commands, you can jump back and forth between the help window and the text window and finally remove the help window again. If, on the other hand, the help text fills the entire page, you can use Ctrl+X, B, Enter to return to your actual text. Internally, the management of multiple texts—for example, your text and the help text—is handled by buffers (Section 13.5).

Keyboard Shortcut	Function
F1, F1	Overview of existing help commands
F1, B	Overview of all keyboard shortcuts (bindings)
F1, C *keyboard_shortcut*	Short description of the assigned command (command)
F1, F *command* Enter	Short description of the command (function)
F1, Shift+F	Emacs FAQs
F1, I	Starts the info system for displaying hierarchical help texts (for operation, see Chapter 6, Section 6.3)
F1, T	Introduction to using Emacs (tutorial)
F1, Ctrl+F *name*	Starts the info system and displays information about the specified command

Table 13.2 Using Online Documentation

13.1.3 Editing Modes

Emacs provides different editing modes in which additional commands are available for editing special files. A distinction is made between the main and secondary modes: only one main mode can be active at a time. However, it can be supplemented by several submodes.

The most important main modes include those for almost all common programming languages (C, C++, Java, etc.) and the LaTeX mode for editing LaTeX files. When loading a file, Emacs automatically activates the mode that seems appropriate to it (e.g., C mode if the file name ends in `.c`). If Emacs can't detect a suitable mode, it selects the *fundamental* mode as the default.

The most important secondary modes include the *fill* mode for editing continuous text with paragraphs across multiple lines and the *abbrev* mode for the automatic resolution of abbreviations.

If you want to disable peculiarities of Emacs due to a certain mode (such as the automatic indentation of program lines in C mode), you can simply switch to the fundamental mode by using Alt+X `fundamental-mode` Enter. For more detailed information on editing modes, see Section 13.6.

13.1.4 Keyboard

In general, there are three ways to enter Emacs commands: the menu, using keyboard shortcuts (mostly a combination with Ctrl or Alt), or typing the entire command name. The third variant is introduced with Alt+X, such as Alt+X `delete-char` Enter.

The input of commands and other parameters is facilitated by two mechanisms:

- During input, you can add the command name with Tab as you would when entering commands in the shell terminal. Emacs distinguishes between uppercase and lowercase letters.
- You can return to commands previously entered using Alt+X (after introducing the new command with Alt+X) using Alt+P (*previous*) and Alt+N (*next*).

In this book, the key sequences are provided as they would be entered on a keyboard. A plus sign means that several keys must be pressed simultaneously, while a comma indicates that the keys are pressed one after the other. Note that Emacs distinguishes between Ctrl+X, Ctrl+B and the similar looking combination Ctrl+X, B! So it does matter how long you hold down the Ctrl key.

In the Emacs documentation, keyboard shortcuts are presented somewhat differently: DEL refers to Backspace! C represents *control* (meaning Ctrl) and M stands for *meta* (meaning Alt).

The meta key

There is no direct equivalent of the meta key on a standard PC keyboard. M-x can be emulated on a PC keyboard in two ways: by Esc and X (in succession) or by Alt+X. In this chapter, I use the Alt key combination in each case. With some Emacs-compatible programs or when you use Emacs in a text console, there are sometimes problems with the Alt key. Instead of Alt+X, you have to use Esc, X there.

Unicode characters

Unicode characters that are not available on the keyboard can be entered by their hexadecimal code. To do this, you need to execute Alt+X ucs-insert n or, shorter, Alt+X, 8, Enter hexcode. For example, you can enter the euro sign (€) using Alt+X, 8, Enter 20ac. The ucs-insert command also accepts the names of Unicode characters. Thus, Alt+X, 8, Enter euro sign also inserts the euro sign.

13.2 Cursor Movement

Besides the cursor keys, Emacs uses numerous keyboard shortcuts for the cursor movement.

The most important shortcuts are listed in Table 13.3.

Keyboard Shortcut	Function
Alt+F/Alt+B	Moves the cursor one word forward or backward (forward/backward)
Ctrl+A/Ctrl+E	Places the cursor at the beginning or the end of the line
Alt+A/Alt+E	Places the cursor at the beginning or the end of the paragraph
Ctrl+V/Alt+V	Moves the text one page down or up
Alt+</Alt+ Shift+>	Moves the cursor to the beginning or end of the text
Ctrl+L	Scrolls the text so that the cursor is in the center of the screen
Alt+G *n* Enter	Places the cursor in line *n*

Table 13.3 Cursor Movement

Keyboard Shortcut	Function
Ctrl+X, R, space z Enter	Saves the current cursor position in register z
Ctrl+X, R, J z Enter	Jumps to the position stored in register z

Table 13.3 Cursor Movement (Cont.)

Executing cursor commands multiple times

Emacs is able to execute any command several times in a row. To do this, you must first type Alt+*n*, where *n* represents any number. The digits must come from the alphanumeric keyboard section (not from the number pad). You must hold down Alt during the entire number input. Then specify the command you want to use. For example, Alt+*n*, Page↓ scrolls the text down by *n* pages.

This method can also be used to enter text characters. For example, Alt+60, - draws a line.

Saving cursor positions in registers

In a longer text, you may often wish to be able to quickly jump back and forth between different places in the text. For this purpose, the current cursor position can be stored with a command in a so-called register (see the penultimate line in Table 13.3). A register is a memory location that is identified by a text character (letter or number). At a later time, you can then jump back to the originally saved location by specifying this register. Note that registers are not saved when you exit Emacs.

13.3 Editing Text

Selecting, deleting, and inserting text

You have already become familiar with the Del or Ctrl+D keys and Backspace to delete individual characters. To delete larger chunks of text, you can use the commands listed in Table 13.4. If you execute the delete commands listed there multiple times in immediate succession, Ctrl+Y will reinsert the entire deleted text. Ctrl+Y can be executed multiple times and at any position in the text. The command thus allows you to move or copy the deleted text to another place.

Keyboard Shortcut	Function
Alt+D	Deletes the next word
Alt+Backspace	Deletes the previous word or the beginning of the word up to the cursor
Ctrl+K	Deletes the end of the line from the cursor position

Table 13.4 Deleting and Reinserting Text

Keyboard Shortcut	Function
Alt+O, Ctrl+K	Deletes the beginning of the line before the cursor position
Alt+M	Deletes the next paragraph
Alt+Z, *x*	Deletes all characters up to the next occurrence of x (the x character gets deleted as well)
Ctrl+Y	Reinserts the most recently deleted text

Table 13.4 Deleting and Reinserting Text (Cont.)

The commands in Table 13.4 are relatively inflexible because the amount of text to be deleted is rigidly specified. If you want to delete any text excerpt, you must first select it. To do this, you can either use the cursor keys together with Shift, or first set an invisible marker point at the beginning or end of the range by using Ctrl+Space (see Table 13.5). From now on, the selected range is the text between the selected point and the current position of the text cursor.

Keyboard Shortcut	Function
Ctrl+Space	Sets an (invisible) marker point
Ctrl+W	Deletes the text between the marker point and the current cursor position
Ctrl+Y	Reinserts the most recently deleted text
Ctrl+X, Ctrl+X	Swaps cursor position and marker point

Table 13.5 Selecting Text

CUA mode

Using Alt+X `cua-mode`, you can activate the common-user-access mode. If you have selected text using Shift, you can copy it to the clipboard by using Ctrl+C or cut it using Ctrl+X as in almost any other program. Ctrl+V inserts the current clipboard content at the cursor position. If no text is selected, Ctrl+X initiates various Emacs commands as before.

Insert and overwrite mode

Emacs is usually in insert mode. That is, newly entered text is inserted into the existing text at the current cursor position. If you want to overwrite the existing text instead, you need to switch to overwrite mode via Alt+X `overwrite-mode` Enter or by using Ins. Executing the command again switches the mode off again.

Correcting common typos

A common typo occurs when two letters are swapped. You can use Ctrl+T to easily correct such swaps. The cursor must be positioned on

the second of the two letters in question; for example, on *w* in the word *sawp*.

Similarly, you can use Alt+T to swap two words. If the cursor is positioned at the beginning of a word, this word will be swapped with the previous one. If, on the other hand, the cursor is somewhere in the word, then the word is swapped with the following word. Running Alt+T multiple times causes the first of the two words to move further and further forward. Finally, Ctrl+X, Ctrl+T enables you to swap the current line with the previous line. A repeated execution of the command causes the line above the cursor to slide further and further down.

13.3.1 Tabs

By default and when you edit a normal text, Tab inserts a tab character. Tabs are not visible. You will not notice whether there is a tab character or multiple spaces at a position until you move the cursor over it. With tabs, the cursor moves in jumps.

If you edit program code, Tab causes the beginning of the line to be indented according to the program structure. In LaTeX mode, the Tab key has no effect at all. So if Tab doesn't work as you expect, it's mostly the edit mode that is to blame. To enter a tab character anyway, you must run Ctrl+Q, Tab.

Tab width

Typically, the tab width is eight characters. But you can use Alt+X `set-variable tab-width` to set a different tab width. If you generally want to work with four instead of eight characters per tab, you can also make this setting in the `~/.emacs` configuration file.

Switching between spaces and tabs

In some editing modes, Emacs automatically replaces long sequences of spaces with tabs. The Alt+X `tabify` commands allow you to replace all blank character series with tab characters in the previously selected area. Alt+X `untabify` works the other way round and replaces tabs with a sufficient number of spaces.

indent-tabs-mode

If you include the following line in the `.emacs` configuration file, Emacs generally inserts spaces instead of tab characters:

```
(setq-default indent-tabs-mode nil)
```

13.3.2 Indenting and Outdenting Text Manually

Text indenting and outdenting is required especially in program listings to structure the code. The most important command can be called via Ctrl+X, Tab (see Table 13.6). This command indents the text between

the marker point (Ctrl+Space) and the current cursor position by one space. If you execute Alt+*n* before this command, the selected text area will be indented by *n* characters. A preceding Esc, - moves the text out instead of in.

Keyboard Shortcut	Function
Ctrl+Space	Sets a marker point
Ctrl+X, Tab	Indents text between the marker point and cursor position by one character
Esc, -, Ctrl+X, Tab	Outdents text by one character
Alt+*n*, Ctrl+X, Tab	Indents text by *n* characters
Esc, -, Alt+*n*, Ctrl+X, Tab	Indents text by *n* characters

Table 13.6 Indenting and Outdenting Text

Rectangle commands

If you want to insert or delete rectangular text blocks within lines (e.g., when editing tables or to indent or outdent comments at the end of program lines), you must use the rectangle commands (see Table 13.7). All characters in the area between the marker point and the cursor position are considered to be rectangles. Rectangular text blocks can be edited even more comfortably in CUA mode. When this mode is active, you can start rectangle marking by using Ctrl+Enter. Then you can delete characters in all lines using Del or insert new text with all other keys.

Keyboard Shortcut	Function
Ctrl+Space	Sets a marker point
Ctrl+X, R, O	Opens a rectangular area (rectangle open)—that is, inserts spaces into the rectangular area
Ctrl+X, R, K	Deletes a rectangular area (rectangle kill)
Ctrl+X, R, Y	Inserts a deleted rectangular area at the cursor position (rectangle yank)
Alt+X `string-rectangle`	Inserts a text before each line of the range
Ctrl+Enter	Rectangle marker in CUA mode

Table 13.7 Rectangle Commands

13.3.3 Continuous Text

To this point, I have assumed that you use Emacs to edit program code, configuration files, and so on. But dealing with Emacs is a little different when you want to edit continuous text. Emacs does not normally perform automatic wrapping. If lines are longer than the screen or window width, then a \ character is displayed at the left end and the text continues on the next line.

If you want to break a single longer line, you need to execute the Alt+Q command: this replaces spaces with line breaks in appropriate places. Thus, one long line becomes multiple short lines.

Emacs considers all lines that are not explicitly separated from other lines by a completely blank line as one paragraph. For a program listing, the consequences of this command are of course fatal! Perform an undo operation using Ctrl+X, U.

> **Line Breaks in LaTeX Documents**
>
> If you edit TeX or LaTeX files, a special feature applies to Alt+Q: Lines beginning with a \ character are considered paragraph boundaries and do not break. To perform a wrap anyway, you have to manually join this line with the previous line and execute Alt+Q again. It is even more convenient to install and activate the AUC-TEX package: then Emacs understands LaTeX better and performs the line break more intelligently.

Automatic line break

When you enter new text, it is of course annoying to constantly press Alt+Q. For this reason, a separate continuous text mode exists, which can be activated using Alt+X `auto-fill-mode` Enter (see Table 13.8).

Keyboard Shortcut	Function
Alt+Q	Performs a manual line break
Alt+X `auto-fill-mode` Enter	Activates the continuous text mode (automatic line break)

Table 13.8 Wrapping Continuous Text

When Emacs is in this mode, all new input is automatically wrapped. Existing text isn't changed by this mode. Even deleting text doesn't lead to an automatic wrap, which is why a manual wrap via Alt+Q often has to be forced after changes in an already existing continuous text. The wrap typically occurs after 70 characters at the latest. You can change the wrap

column using the following command: Alt+X set-variable Enter fill-column Enter *n* Enter.

Shortcuts

A special feature of Emacs is that you can use keyboard shortcuts without any preliminary work. To do this, you just enter the first letters of a word and press Alt+/. Emacs then searches first in the preceding text, then in the following text, and finally in all open files for words that begin with these characters. If you type env Alt+/ at this point in the text, Emacs replaces *env* with *environment*. If you press Alt+/ more often, Emacs offers other possible additions, such as *envelope* and *envy*.

Dynamic extensions work only if a word is already in the text of a loaded file (it doesn't have to be the current file) and if the initial letters match.

13.4 Search and Replace

The fastest way to find text is to use Ctrl+S *searchtext*. The command starts the search immediately after the first character has been entered. So if you want to search for secondary mode and type Ctrl+S "sec", the cursor will already jump to the first word that starts with *sec*. Instead of typing the other letters, you can then jump to the next word that also starts with *sec* by pressing Ctrl+S again. (If you enter only lowercase letters, there is no distinction between uppercase and lowercase.) If you now suddenly decide that you actually want to search for severity, you can delete the *c* by using the Backspace key. Emacs then jumps back to the first word (starting from the position at the beginning of the search) that begins with *se*. If you then type "v", Emacs jumps further to the first word that starts with *sev*.

Keyboard Shortcut	Function
Ctrl+S	Incremental search forward
Ctrl+R	Incremental search backward
Alt+P	Selects a previously used search text (*previous*)
Alt+N	Selects a search text to be used later (*next*)
Ctrl+G	Cancels the search
Ctrl+X, Ctrl+X	Swaps the marker point (start of the search) and the current cursor position
Ctrl+Alt+S	Incremental pattern search forward

Table 13.9 Search and Replace Commands

Keyboard Shortcut	Function
Ctrl+Alt+R	Incremental pattern search backward
Alt+%	Search and replace without pattern
Alt+X query-replace-r Enter	Search and replace with pattern

Table 13.9 Search and Replace Commands (Cont.)

As soon as you press Enter or a cursor key, the program assumes that the search is complete and places the cursor at the position it has found. The start of the search is saved by a marker point. For this reason, you can use Ctrl+X, Ctrl+X to easily move the cursor back to where it was at the start of the search. Repeating this Ctrl+X, Ctrl+X pattern will take you back to the place of the search text.

By pressing Ctrl+S twice, you can resume the search and jump to the next occurrence of the search text. If you want to search backward, just press Ctrl+R instead of Ctrl+S.

If at a later time you want to search for a text that you have already searched for previously, you can select a text from the saved list of search texts using Alt+P (*previous*) and Alt+N (*next*) after pressing Ctrl+S.

13.4.1 Searching for Patterns (with Regular Expressions)

The incremental search finds texts that exactly match the search text. But in many cases, it's preferable to search for texts that correspond to a certain pattern. You can start such a search via Ctrl+Alt+S or +R.

Search Pattern	Function
\<	Beginning of a word
&	End of a word
^	Beginning of the line
$	End of the line
.	Any character (except the line break)
.*	Any number (including 0) of arbitrary characters (like * in filenames)

Table 13.10 Structure of Regular Search Pattern

Search Pattern	Function
`.+`	Any number of arbitrary characters (but at least one)
`.?`	None or any character
`[abc..]`	One of the listed characters
`[^abc..]`	None of the listed characters
`\(`	Start of a group (see Section 13.4.2)
`\)`	End of a group
`\x`	Special character *x* (e.g., `\\` to search for a `\` character or `\.` to search for a dot).
`\&`	Placeholder in the replace pattern for the entire text that has been found
`\1`	Placeholder in the replace pattern for the first `\(...\)` group in the search text

Table 13.10 Structure of Regular Search Pattern (Cont.)

The search text is case-sensitive. Here are some examples to show the syntax of the pattern string (see Table 13.10):

- `\<[Tt]he\>` searches for the article *the*, whether it is lowercase or uppercase. Compound words that contain *the* (such as *thesis*) are ignored.
- `[Tt]he[a-z]+` searches for compound words beginning with *The* or *the* followed by at least one other letter. The cursor stops at the end of each word (at the first character that is not a letter between *a* and *z*).

The character pairs `\(` and `\)` have no influence on the actual search. However, the characters in the searched text that correspond to the characters contained in the group can then be reused to form the replace text (see the next section).

13.4.2 Search and Replace

Emacs also distinguishes between the normal command and the extended version with pattern search when searching and replacing. The normal variant with Alt+% ignores uppercase and lowercase during the search. When replacing (see Table 13.11), the initial letters of words remain as they were before if the replace text is completely lowercase. The find and replace command cannot be used for multiline texts because the wildcard characters * and + are not effective beyond one line.

Keyboard Shortcut	Function
Space or Y	Replace, continue search.
,	Replace, but leave the cursor so that the result can be checked. If everything is OK, the command can be continued by pressing the spacebar.
Backspace or N	Do not replace, continue search.
Esc	Do not replace, cancel command.
!	Perform all further replacements without confirmation.
Ctrl+R	Temporarily interrupt the command to make a manual correction at the cursor position (recursive edit).
Ctrl+Alt+R	Resume replace command.

Table 13.11 Keyboard Shortcuts for Editing Found Text

You can start the search and replace process with patterns using Alt+X `query-replace-r` Enter. In the replace string, you can use `\&` and `\`*n* to specify placeholders that correspond to the entire search pattern or part of it (see Table 13.10). This allows very complex operations to be carried out efficiently.

The following example illustrates this: You replace `function(\([^,]*\),\([^,]*\))` with `funktion(\2,\1)`. Each time `function` is called, the two parameters are swapped. Therefore, `function(a+b,2*e)` becomes `function(2*e,a+b)`. The only condition is that there must be no commas in the parameters of the function. If the parameters in `function(f(a,b), g(x,y))` are swapped, the command fails.

You should use the search and replace command with patterns with caution and save your text first. Especially during the first attempts, it often happens that the search pattern captures completely different (often much larger) texts than you planned. Ctrl+X, Ctrl+U undoes incorrect replace commands if necessary.

13.5 Buffers and Windows

When editing multiple texts, Emacs manages each text in a buffer. Even if you work with only one text, there are multiple buffers: one for the text (the name of the buffer matches its file name), one for an info or help window that was opened at some point (buffer name `*info*` or `*help*`), one for

the last displayed list of possible commands that were completed by Tab (*completions*), and so on.

Besides the term *buffer*, Emacs also uses the term *window*; a window is an area within Emacs where a buffer is displayed.

Typically, only one window is used, which uses the entire available space. When multiple commands are executed (e.g., to display help or other internal Emacs information), the screen will be split horizontally into two windows. It can also be divided into multiple horizontal or vertical strips. Each area (window) can then display a different buffer.

It's also possible to display the same buffer in two windows. This is especially useful for very long texts as you can edit two different sections of the text without having to constantly move the cursor.

The commands in Table 13.12 refer to the currently active window (i.e., the window in which the cursor is located). The commands change the buffer displayed in this window.

Keyboard Shortcut	Function
Ctrl+X, B, Enter	Activates the previously used buffer.
Ctrl+X, B, *name* Enter	Activates the specified buffer.
Ctrl+X, Ctrl+B	Displays the list of all possible buffers in a window. This window can be deleted again using Ctrl+X, 1.
Ctrl+X, K *name* Enter	Deletes the specified buffer. If the buffer contains a file that hasn't yet been saved, a confirmation prompt appears.

Table 13.12 Buffer Commands

The commands in Table 13.13 only affect the display of buffers in different screen areas (windows). The dividing line between the windows can be moved with the mouse. The buffers are not affected by deleting a window. They become invisible but remain in the memory and can be displayed again at any time.

Keyboard Shortcut	Function
Ctrl+X, O	Jumps to the next window (letter O)
Ctrl+X, 0	Deletes the current window (zero)
Ctrl+X, 1	Deletes all windows except the one where the cursor is

Table 13.13 Window Commands

Keyboard Shortcut	Function
Ctrl+X, 2	Divides the current window into two horizontal areas
Ctrl+X, 3	Divides the current window into two vertical areas
Ctrl+X, <	Moves the window content to the left
Ctrl+X, >	Moves the window content to the right

Table 13.13 Window Commands (Cont.)

Different Types of Windows

The term *window* in Emacs has nothing to do with a conventional window on GNOME or KDE, but refers only to a subarea within the Emacs window. If you actually need a second Emacs window—for example, to conveniently edit two program listings side by side—you should run **File • New Frame**.

13.6 Special Editing Modes

Numerous editing modes change the functionality of the editor and provide additional special commands. This way, Emacs is adapted perfectly to a text type. Depending on the mode, keywords and comments are also highlighted in color. Emacs distinguishes between major modes and minor modes (see Table 13.14 and Table 13.15).

Keyboard Shortcut	Function
Alt+X fundamental-mode Enter	Standard mode (default setting)
Alt+X text-mode Enter	Mode for convenient text indentation
Alt+X c-mode Enter	C mode
Alt+X c++-mode Enter	C++ mode
Alt+X emacs-lisp-mode Enter	Edit Emacs Lisp files (e.g., ~/.emacs)
Alt+X html-mode Enter	HTML mode
Alt+X java-mode Enter	Java mode
Alt+X latex-mode Enter	LaTeX mode
Alt+X sh-mode Enter	Mode for editing shell scripts

Table 13.14 Major Emacs Modes

Keyboard Shortcut	Function
Alt+X auto-fill-mode Enter	Continuous text mode (automatic word wrap)
Alt+X cua-mode Enter	CUA mode (Ctrl+C, Ctrl+X, and Ctrl+V)
Alt+X font-lock-mode Enter	Colored syntax marking
Alt+X iso-accents-mode Enter	Input of special characters for foreign languages
Alt+X abbrev-mode Enter	Abbreviation mode (automatic resolution of abbreviations)

Table 13.15 Minor Emacs Modes

Only one main mode can be active at a time. This mode is automatically selected according to the filename identifier and keywords in the text. The major mode can be supplemented by minor modes. Each file that is edited in Emacs (for each buffer) has its own mode setting. The manual change of the mode always affects only the current buffer. Changing to another main mode deactivates the previous mode. Turning a minor mode on or off does not change the major mode.

F1, A mode Enter provides an overview of all available modes. Information about the currently active main mode can be obtained by pressing F1, M.

Syntax highlighting

Perhaps the most attractive feature of the editing modes is the syntax highlighting. You can use this feature to highlight commands, comments, and so on by colors or font attributes. This makes program code, LaTeX documents, and the like much clearer.

PHP and markdown mode

Inexplicably, Emacs does not feature a PHP mode by default. A mode for editing markdown documents is also missing. At the end of Section 13.7.4, you'll discover how you can easily install these and other Emacs extensions.

Automatic and Explicit Mode Setting

When loading a file, Emacs tries to detect its file type and then activates the adequate mode. If that does not work, you can activate the mode manually as described previously. Alternatively, you can add a comment to the first line of the file containing the characters *-* modusname *-*—for example, *-* html *-*.

13.7 Configuration

This section provides an overview of the Emacs configuration files and describes some elementary configuration steps.

Personal configuration

The user-specific configuration can be performed either by using menu commands (**Options** menu) or by a direct modification of the `.emacs` configuration files. Note that some distributions automatically copy Emacs configuration files from `/etc/skel` to the user directory when creating new Linux users!

Global configuration

If your version of Emacs behaves differently than described in this book, the configuration of your Linux distribution is often responsible. The settings can be located in both personal and global configuration files (`site-start.el`):

```
/usr/share/emacs/site-lisp/site-start.el
/usr/share/emacs/site-lisp/debian-startup.el      (Debian, Ubuntu)
/usr/share/emacs/site-lisp/site-start.d/*         (Red Hat, Fedora)
/usr/share/emacs/site-lisp/site-start.el          (SUSE)
```

13.7.1 Setting the Font

Let's start with two important keyboard shortcuts: Ctrl+X, Ctrl++ enlarges the default font, while Ctrl+X, Ctrl+- reduces it. So if you want to increase your reading comfort or to get more space to display particularly long lines, these key combinations will get you there quickly.

To permanently reset the default font, you need to run **Options • Set Default Font**. You can then set the relevant font in a dialog. Note that this setting is not saved to `.emacs` until you run **Options • Save Options**.

13.7.2 Configuration at the Click of a Mouse

All advanced customization options, of which there are thousands, are accessible via **Options • Customize Emacs • Top-Level Customization Group** (see Figure 13.2).

The buttons on this page lead to more dialogs for different groups of options. On each page, you can use buttons to save all changes made either only until the program terminates or permanently in `.emacs` (**Set for Current Session** or **Save for Future Sessions**). Unfortunately, the dialogs are deeply nested. It isn't always easy to find the right dialog for a particular option.

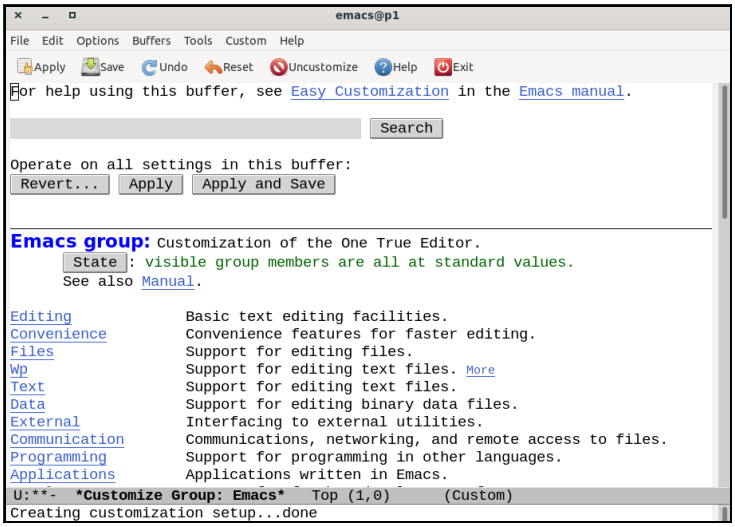


Figure 13.2 Emacs Configuration

13.7.3 Manual Configuration Directly in .emacs

Example of .emacs

Instead of clicking through nested dialogs, you can also perform the configuration directly in `.emacs`. There isn't enough space here for a detailed description, but the following commented listing provides some examples of commonly used options and settings. Depending on the Linux distribution, some of these settings apply by default:

```
; file .emacs
; no welcome screen
(setq inhibit-startup-message t)

; show row and column number in the status bar
(line-number-mode 1)
(column-number-mode 1)

; make selected text area visible
(setq-default transient-mark-mode t)

; use <Tab> to insert spaces instead of tabs
(setq-default indent-tabs-mode nil)

; end last line automatically with newline code
(setq require-final-newline t)

; save cursor position
(require 'saveplace)
```

```
(setq-default save-place t)

; activate AUC TEX (this is an extended LaTeX mode;
; the auctex package must be installed separately; the package
; name is emacs-auctex for many distributions
(require 'tex-site)

; highlight associated bracket
; https://www.emacswiki.org/emacs/ShowParenMode
(show-paren-mode 1)

; store all backup files (name~) in *one* directory
; https://www.emacswiki.org/emacs/AutoSave
(setq backup-directory-alist
  `(("." . ,(concat user-emacs-directory "backups"))))
; custom function; swaps the letter
(defun swap-char()       ;two letters at the cursor position
  (interactive)          ;swap:
  (save-excursion
  (forward-char)
  (transpose-chars 1)))

; a few custom keyboard shortcuts
(global-set-key [f2]  'switch-to-buffer)     ;switch buffers
(global-set-key [f3]  'goto-line)            ;go to line n
(global-set-key [f4]  'advertised-undo)      ;undo function
(global-set-key [f5]  'swap-char)            ;swap letters
```

Emacs Lisp programming

You have even more configuration options if you engage in Emacs Lisp programming. Doing so allows you to program your own commands in the `.emacs` configuration file and then associate them with keyboard shortcuts.

13.7.4 MELPA

MELPA (*https://melpa.org*) stands for Milkypostman's Emacs Lisp Package Archive and is probably the best maintained archive of Emacs extensions. MELPA is preconfigured in current Emacs versions. You can access the central package management dialog using **Options • Manage Emacs Packages** or Alt+X `list-packages`.

As usual, you can quickly search the text with Ctrl+S. Clicking a package displays detailed information and an **Install** button. Some of my favorites are the `php-mode` and `markdown-mode` extensions. They make editing PHP and markdown files much easier.

PART V

System Configuration and Administration

Chapter 14
Basic Configuration

This chapter is the first of a whole series of chapters on Linux system configuration. The focus is on rather elementary functions:

- Text console configuration
- Setting the date and time
- User administration
- Internationalization, character set, Unicode
- Overview of the hardware configuration
- CPU tuning
- Notebook optimization (energy saving measures, battery charging behavior, etc.)
- CUPS printing system
- Logging with syslog and journal
- Cockpit (web console for system administration)

The other chapters then cover topics like network configuration, package management, system library management, configuring the graphics system (X and Wayland), file system administration, system startup (GRUB, systemd), and dealing with the kernel and its modules.

14.1 Introduction

This and the following chapters provide a behind-the-scenes look at Linux configuration programs. You will learn what is controlled and preconfigured where and how. For this reason, you will find a lot of background information about how the overall system works.

Unfortunately, the various distributions differ in many small details when it comes to configuration. In this book I try to describe the common denominator of as many Linux systems as possible. Nevertheless, it can happen that your distribution may approach some individual details a little differently.

In those cases, you'll need to consult the documentation or search the internet. Even if this may be inconvenient for you from time to time, I'm still convinced that the general approach is a better one than a book on Red Hat administration, another one on Ubuntu administration, and so on—

not least because the individual distributions also change from version to version. So, sooner or later, you will have to learn to read and understand the various manuals, help pages, and the like. This book is intended not to replace original manuals or step-by-step instructions, but to provide basic knowledge!

Configuration and admin tools

Some distributions offer convenient configuration programs that can be used both during and after the installation. SUSE with YaST particularly stands out here. Desktop systems such as KDE and GNOME also contain configuration tools whose effects extend beyond the desktop.

These tools should always be the first choice in case of basic configuration issues!

In addition to the configuration programs supplied, there are also external tools or Linux distributions of their own with additional admin tools. Many of them can be operated via a web interface. Table 14.1 provides an overview of some popular tools.

URL	Function	Is It Free?
https://cockpit-project.org	System and network administration	•
https://webmin.com	System and network administration	•
https://fai-project.org	Software distribution and installation	•
https://m23.sourceforge.io	Software distribution and administration	•
http://directory.fedoraproject.org	LDAP user interface for RHEL/Fedora	•
https://sso.redhat.com	Red Hat administration	
https://ubuntu.com/landscape/docs	Ubuntu administration	
https://www.univention.com/products/ucs/	LAN and mail server configuration	
https://zentyal.org	LAN server configuration	
https://cpanel.net	Root and web server administration	
https://plesk.com	Root and web server administration	

Table 14.1 Selected Administration Tools

The purpose and functionality of these tools varies greatly. Some of the tools listed are specifically intended for maintaining a large number of similar Linux installations; others are for server configuration only. In this book, I will only introduce the Cockpit program in more detail (see Section 14.14).

A bullet in the Is It Free? column means that it's open-source software that can also be used free of charge by companies. Most commercial products also have free variants with reduced functionality.

Avoid Dependencies!

Using commercial administration tools, you can easily drift into new dependencies. Also, I've seen a lot of administration tools come and go in the past. Do not lightly opt for external administration tools!

Configuration files

Sophisticated configuration tools with nice user interfaces take the hassle out of modifying Linux configuration files directly. Especially for Linux beginners, this is undoubtedly very useful. However, there are a number of reasons to take a closer look at the configuration files and thus the internal workings of Linux:

- The configuration files can be modified with any text editor, even in a text console or via an SSH connection.
- Once you understand how to configure a particular Linux feature, you can apply that knowledge to almost any other Linux distribution.
- You can only control all aspects of a system function if you change the configuration files directly. Configuration tools, on the other hand, are often limited to a few particularly important details.
- It's easy to copy configuration files from one computer to another. This can save a lot of time when you change distributions, reinstall Linux on another computer, and so on.
- The better you understand the structure of the configuration files and the control options they provide, the better you will understand Linux, and the less your computer will be the proverbial black box that nobody can see into.
- Simple configuration files can be managed pretty well in a version control system like Git. You can use it to track changes and clone configurations to different systems.

The End of the Line Can Be Crucial!

When editing configuration files, be sure to end the last line using Enter. Some Linux programs do not process files correctly if the end of line is missing in the last line.

/etc directory

Almost all Linux configuration files are located in the /etc directory. For this reason, a reference of all configuration files discussed in this book can be found in the index under the letter E.

Related configuration files of larger programs are often organized in their own subdirectories (see Table 14.2).

Directory	Contents
/etc/cron*	Cron files (see Chapter 10, Section 10.6)
/etc/default	Distribution-specific files (Debian, Ubuntu)
/etc/init.d, /etc/rc*.d	Init-V system (see Chapter 20, Section 20.4)
/etc/systemd	Systemd (see Chapter 20, Section 20.1)
/etc/sysconfig	Distribution-specific files (Fedora, Red Hat, SUSE)
/etc/X11	Graphics system

Table 14.2 Important /etc Directories

/etc backup

It's a good idea to back up the entire /etc directory every now and then. This allows you to quickly determine what the original state of a particular configuration file was at any time after changes have been made:

```
root# mkdir /etc-backup
root# cp -a /etc/* /etc-backup
```

Wonder Weapon: etckeeper

If you are familiar with the Git version control program, then you can use a fantastic tool called etckeeper. This tool backs up the state of all /etc files in a Git repository at each package installation or once per day. If necessary, you can restore individual configuration files from it at any time. The installation instructions can be found here: *https://wiki.archlinux.org/title/Etckeeper*.

Finding configuration files

If you can't find a configuration file in your distribution, this can have several causes: the underlying program packages may not even be installed, or the configuration files may be in a different location in your distribution. You can use the `locate`, `find`, and `grep` commands for searching. The following command shows how you can search `/etc` and all subdirectories for files whose contents (not the filename) contain the word `abcde` in any case:

```
root# cd /etc
root# grep -r -i abcde
```

Activating a new configuration

For some programs, changes to the configuration files don't take effect until you restart the program or explicitly ask it to reread the configuration files. For this purpose, `systemctl reload` is provided. A few services do not support `reload`, in which case you must use `restart` instead:

```
root# systemctl reload  servicename
root# systemctl restart servicename
```

In contrast to Windows, it's almost never necessary to restart the computer. The only exceptions are changes to the kernel and some hardware-specific settings that can only be made immediately at system startup.

14.2 Configuration of the Text Consoles

In modern distributions, Linux starts the graphics system directly, and Linux beginners often don't even know that text consoles exist at all. From time to time, however, it happens that the configuration for the graphics system is faulty or that no graphics system is available for other reasons. In server installations or in virtual machines, the graphics system is often deliberately omitted. In those cases, you'll have to get used to the text consoles. For elementary settings like keyboard layout and font, either the `kbd` system or the newer `console` system is responsible, depending on the distribution. In detail, the configuration looks a little different for each distribution.

14.2.1 Keyboard Layout

Debian/Ubuntu

On Debian and Ubuntu, the programs of the `console-setup` package are responsible for the keyboard layout. The `/etc/default/keyboard` configuration file controls the keyboard layout:

```
# /etc/default/keyboard
# keyboard
XKBMODEL="pc105"
XKBLAYOUT="en"
XKBVARIANT=
XKBOPTIONS="lv3:ralt_switch"
```

Whether you're using Debian or Ubuntu, if possible, you should make changes to the keyboard layout not by directly modifying the configuration files, but by using the following command instead:

```
root# dpkg-reconfigure keyboard-configuration
```

The analysis of the files is taken care of by systemd (`keyboard-setup` service), which executes the `/lib/console-setup/keyboard-setup.sh` script.

Arch, Fedora, and RHEL

In Arch Linux, Fedora, and RHEL, the `kbd` package and systemd are responsible for setting the keyboard layout. The configuration is stored in two files—in `/etc/vconsole.conf` for text mode and in `/etc/X11/xorg.conf.d/00-keyboard.conf` for the X graphics system:

```
# file /etc/vconsole.conf
KEYMAP="en"
FONT="eurlatgr"
```

The `localectl` command `list-keymaps` returns a list of all possible keyboard layouts. You can use `localectl set-keymap` for configuration purposes. `localectl` tries to apply the setting to the graphics mode as well, but this isn't always successful. In the following example, all German keyboard layouts are determined first, and then the variant for Apple's Mac keyboard is set:

```
root# localectl list-keymaps | grep '^de'
de-deadacute
de-mac
...
root# localectl set-keymap de-mac
```

On Fedora and RHEL, the systemd service `systemd-localed` is responsible for the configuration. On Arch, `systemd-vconsole-setup` takes care of it.

SUSE

SUSE also uses the `kbd` package. The configuration is located in `/etc/sysconfig/keyboard` and is analyzed by systemd (`systemd-vconsole-setup` service). As with Red Hat systems, the keyboard settings for X are located in the `/etc/X11/xorg.conf.d/00-keyboard.conf` file created by `systemd-consoled`.

14.2.2 Font

Consoles are basically Unicode-compatible. However, the maximum number of possible characters in console fonts is very small (256 or 512). For this reason, console fonts can only ever represent a tiny fraction of all Unicode characters.

Debian/Ubuntu

The configuration settings are located in `/etc/default/console-setup`. The file is analyzed by the `console-setup` systemd service:

```
# /etc/default/console-setup (Debian default settings)
...
# Font
CHARMAP="UTF-8"
CODESET="Lat15"
FONTFACE="fixed"
FONTSIZE="8x16"
```

Fedora and RHEL

Arch, Fedora and RHEL store the desired console font in the `/etc/vconsole.conf` file (`FONT` variable). This file is analyzed by systemd (`systemd-vconsole-setup` service).

SUSE

On current SUSE versions, the console font is set in `/etc/sysconfig/console`. Systemd parses this file (`systemd-vconsole-setup` service).

14.3 Date and Time

Due to the international networking of computers, the use of a worldwide standardized time is required. On Linux computers, Coordinated Universal Time (UTC) is the measure of all things or time. This is the successor of the better-known Greenwich Mean Time (GMT).

When you save a file, it saves not the current local time, but a time converted to this international standard. If you then view the file using `ls -l`, the time is converted back to the local time at the computer's location. This procedure makes it possible to determine, for example, which file is more current: a file saved at 6:00 p.m. local time in Munich or a file saved at 12:30 p.m. local time in New York.

Time setting during computer startup

The time setting during the computer startup is mostly done in two phases. First, the mainboard clock is read. Its main drawback is its relatively low accuracy. Once connected to the internet, Linux can synchronize its time with other servers on the internet (see Section 14.4).

To read the hardware clock, often called the *CMOS clock* or *real-time clock* (RTC), you can use the `hwclock` command. In some distributions, this is executed by systemd (`systemd-timedated` service).

Setting the time zone

For commands like `ls` or the file managers of KDE and GNOME to convert GMT time to local time and display it accordingly, each program must know in which time zone it is running. Almost all Linux programs make use of functions of the *glibc library* for this purpose. This library analyzes the `/etc/localtime` file. This file is the link to a time zone file from the `/usr/share/zoneinfo` directory or a copy of this file. The time zone file also contains information on whether a particular daylight saving time scheme applies in a particular time zone (such as Central European Summer Time [CEST]).

The most convenient way to determine the current setting is to use `timedatectl`:

```
user$ timedatectl
               Local time: Wed 2023-04-26 17:42:08 CEST
           Universal time: Wed 2023-04-26 15:42:08 UTC
                 RTC time: Wed 2023-04-26 15:42:08
                Time zone: Europe/Vienna (CEST, +0200)
System clock synchronized: yes
              NTP service: active
          RTC in local TZ: no
```

To change the time zone, you can use `timedatectl list-timezones` to determine all time zones available for selection and then set the relevant time zone using `timedatectl set-timezone`:

```
user$ timedatectl list-timezones | grep Europe
Europe/Amsterdam
Europe/Andorra
Europe/Astrakhan
Europe/Athens
...
root# timedatectl set-timezone Europe/Vienna
```

Behind the scenes, `timedatectl set-timezone` simply changes the link from `/etc/localtime`:

```
root# ln -s -f /usr/share/zoneinfo/Europe/Vienna /etc/localtime
```

In Debian and Ubuntu, the `/etc/timezone` text file also specifies the time zone. However, this file is analyzed not directly by the glibc library, but only by the tools for resetting the `localtime` file.

Configuration tools

Depending on the distribution, desktop or init system, you can use a configuration program to change the time zone and date and time of the computer clock:

- **GNOME**
  System settings, **Date and Time** module
- **KDE**
  System settings, **Regional Settings** module • **Date & Time**
- **Systemd**
  `Timedatectl`
- **Debian, Ubuntu**
  `dpkg-reconfigure tzdata`
- **SUSE**
  YaST module, **System • Date and Time**

14.4 Synchronizing Date and Time via NTP

As soon as a computer has an internet connection, the time is regularly synchronized with time servers on the internet. The *Network Time Protocol* (NTP) is provided for this purpose. NTP makes it possible that the local time on the computer deviates only minimally (in the microsecond range) from the time given by atomic clocks. Another advantage is that the then (almost) perfectly accurate time can also be used to readjust the CMOS clock from time to time.

There are various ways to use NTP:

- In distributions with systemd (see Chapter 20, Section 20.1), the `time-sync` target can be active. This invokes the start of the `systemd-timesyncd` daemon, which takes care of the time setting via NTP.
- The `ntpdate` command obtains the exact time from the internet *once* and sets the clock of the computer. For computers that are switched on and off frequently, this is sufficiently accurate.
- The `ntpd` background service synchronizes the time permanently with external NTP servers.
- This task can also be performed by the modern Chrony program.

Abrupt Time Changes Are Dangerous

There are quite a lot of programs that do not tolerate sudden time changes well. These include, for example, the Kerberos authentication system, the Dovecot POP3 and IMAP servers, and database servers. It is recommended

to shut down and then restart such programs before running `ntpdate` or manually changing the time. Dovecot automatically stops when it detects that the time has been reset.

More information about date and time management can also be found on the following pages:

- *https://wiki.archlinux.org/title/System_time*
- *https://wiki.archlinux.org/title/Network_Time_Protocol_daemon*

14.4.1 The systemd-timesyncd Process (Debian, Raspberry Pi OS, and Ubuntu)

In current Debian and Ubuntu distributions, the `systemd-timesyncd` background process is usually running. It's executed as part of the systemd target, `time-sync` (see also `man systemd.special`).

The `systemd-timesyncd` background service accesses NTP servers that have already been fixed at compile time. You can determine this as follows:

```
root# strings /lib/systemd/systemd-timesyncd | grep ntp
sd_network_get_ntp
ntp.ubuntu.com
property_get_ntp_message
```

Deviating from this, other NTP servers can be named in `/etc/systemd/timesyncd.conf`.

14.4.2 Chrony (Fedora, RHEL, and SUSE)

Fedora and RHEL

On Fedora, the classic NTP tools were replaced years ago by the Chrony program. On RHEL, this step took place with version 8.

Chrony is particularly well-suited for notebooks and virtual machines that are not constantly connected to the internet and whose time often needs to be significantly corrected after longer offline periods. You can find more information on Chrony at *https://chrony.tuxfamily.org*.

SUSE

SUSE Enterprise and openSUSE have also switched to Chrony with version 15. The configuration is done in the YaST **System • Date and Time** module; when the configuration is changed, `ntpdate` is run once. To set up an NTP server yourself, start the YaST **Network Services • NTP Configuration** module and enable the **Start NTP Service Now and at System Startup** option. openSUSE-specific information can be found at *https://doc.opensuse.org/documentation/leap/reference/html/book-reference/cha-ntp.html*.

Configuration

The configuration is done via `/etc/chrony.conf`. By default, the `pool` parameter there causes Chrony to obtain the time from several public NTP servers of the *http://pool.ntp.org* project. Alternatively, you can explicitly specify your own NTP servers via `server` (see `man chrony.conf`):

```
# file /etc/chrony.conf
pool 2.fedora.pool.ntp.org iburst
...
```

If the automatic resetting of the time doesn't work correctly after a long offline period, you should restart Chrony using the following command:

```
root# systemctl restart chronyd
```

Determining the status

Chrony consists of two programs: the `chronyd` background service (daemon) and the `chronyc` command. The latter allows you to set various parameters monito the chrony status. `chronyc help` provides an overview of the available commands. Similar to `ntpq -p`, `chrony sources` summarizes which NTP servers Chrony uses as time source and how big the time deviations are at a given moment:

```
root# chronyc sources

210 Number of sources = 4
MS Name/IP address         Stratum Poll Reach LastRx Last sample
===============================================================================
^* ntp.cnh.at                    2   8   377   458   -129us[ -163us] +/-   49ms
^+ mail.hoco.at                  2   9   377   270   +107us[ +107us] +/-   86ms
^+ 193.170.62.252                2  10   377   134   -421us[ -421us] +/-   62ms
^+ ns5.nosuchhost.net            2   9   377   337  +1123us[+1123us] +/-   42ms
```

14.5 Users, Groups, and Passwords

On Linux, access rights control who can access which files, who can run which programs, and who can access which hardware components or device files (see Chapter 9, Section 9.6). These access rights are based on user administration: By default, various users are set up during the installation of Linux. Each user is assigned to at least one, but possibly several groups. Groups are used to allow multiple users to access common files or programs.

By its very nature, user management provides the option to set up multiple users on one computer, who can then log in and work in isolation from each other. However, user management is also important if you work alone on your Linux computer: For security reasons, many system services are

not executed with `root` privileges, but also in special system accounts with restricted rights. These accounts are blocked for ordinary login and are intended only for the execution of programs.

This section describes which tools help you set up users and groups, and which configuration files are responsible behind the scenes for assigning users, groups, and passwords.

Configuration programs

Basically, you can manage users manually as `root` by directly changing the files described in this section. It's more convenient and secure to use the user and group management tools provided with most distributions:

- **GNOME**
  User system settings module
- **KDE**
  User system settings module
- **SUSE**
  Security • User and Group Management YaST module (see Figure 1.1)

Commands

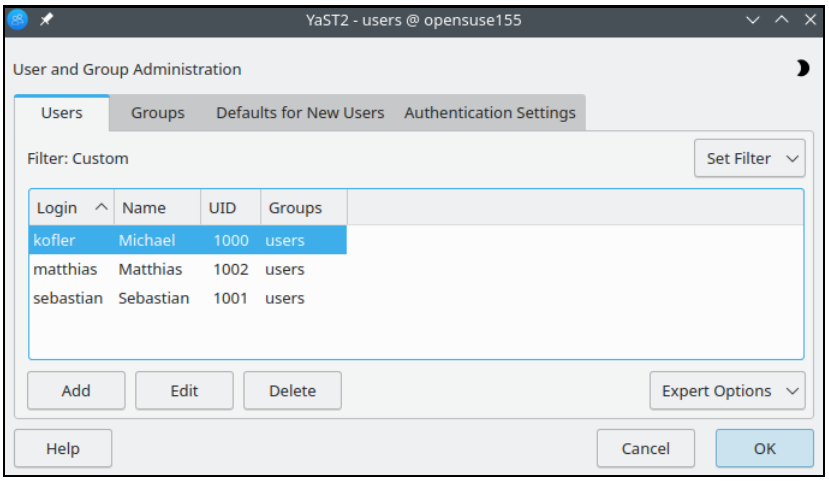


Figure 14.1 User Management in OpenSUSE

If you have to do without comfortable user interfaces or if you want to automate the user administration in scripts, you can use the commands summarized in Table 14.3.

Command	Function
`adduser`	Sets up a new user (Debian)
`addgroup`	Sets up a new group (Debian)

Table 14.3 Commands for User and Group Management

Command	Function
chage	Controls how long a password remains valid
chgrp	Changes the group membership of a file
chmod	Changes the access bits of a file
chown	Changes the owner of a file
chsh	Changes the default shell of a user
delgroup	Deletes a group (Debian)
deluser	Deletes a user (Debian)
groupadd	Sets up a new group
groupdel	Deletes a group
groupmod	Changes group properties
groups	Displays the groups of the current user
id	Displays the current user and group ID number
newgrp	Changes the active group of a user
newusers	Sets up multiple new users
passwd	Changes the password of a user
useradd	Sets up a new user
userdel	Deletes a user
usermod	Changes user properties

Table 14.3 Commands for User and Group Management (Cont.)

The following example shows how you can create the new `testuser` user and assign a password to it:

```
root# useradd -m testuser
root# passwd testuser
New passwd: xxxxxxxx
Re-enter new passwd: xxxxxxxx
```

Dual-track commands

You'll notice that there are two commands for some tasks, such as `adduser` and `useradd`. `Adduser`, `addgroup`, `deluser`, and `delgroup` are Debian-specific extensions to the traditional commands, `useradd`, `groupadd`, and so on. On Debian and Ubuntu, these commands take into account the rules defined in `/etc/adduser.conf` and `/etc/deluser.conf`.

Red Hat and Fedora cause some confusion in this regard: There, the `adduser`, `addgroup`, `deluser`, and `delgroup` commands are also available. However, these are not the commands familiar from Debian, but links to `useradd`, `groupadd`, `userdel`, and `groupdel`. For this reason, `adduser` on Fedora has a different syntax than `adduser` on Debian!

14.5.1 User Management

There are three types of users in Linux:

- **Super user (also known as system administrator or root)**
  This user usually has the name `root`. Anyone who knows the `root` password and logs in as `root` has unrestricted rights: He or she may view, modify, or delete all files, run all programs, and so on. Such a long list of rights is only required for system administration purposes. For security reasons, you should not perform all other tasks as `root`!
- **Ordinary users**
  These users perform their day-to-day work in Linux. They have full access to their own files, but limited access to the rest of the system. The first or last name of the user is often used as the login name, such as `catherine` or `smith`.
- **System users for daemons and server services**
  Finally, there are a number of users that are not intended for interactive work on the computer but are used to run specific programs. For example, the Apache web server is not run by the `root` user but by a separate user called `apache`, `wwwrun`, `httpd`, or something similar, depending on the distribution. This approach is chosen to achieve the highest possible system security.

/etc/passwd

The list of all users is stored in the `/etc/passwd` file. This is where the login name, full name, UID and GID numbers, home directory, and shell are stored for each user. The following format applies:

Login:Password:UID:GID:Name:Home Directory:Shell

The following lines show some user definitions in `/etc/passwd` on Ubuntu Linux:

```
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/bin/sh
bin:x:2:2:bin:/bin:/bin/sh
sys:x:3:3:sys:/dev:/bin/sh
...
kofler:x:1000:1000:Michael Kofler,,,:/home/kofler:/bin/bash
hoover:x:1001:1001:Herbert Hoover,,,:/home/hoover:/bin/bash
```

The passwd name suggests that the passwords are also stored in the file. That was indeed the case in the past. Today, /etc/passwd contains only the *x* character instead of the password information. The encrypted password hash is stored in the separate /etc/shadow file, which I will introduce in Section 14.5.3.

Login name

The login name should consist of lowercase letters only (US-ASCII letters and numbers) and should not be longer than eight characters. Although both non-ASCII characters and more than eight characters are basically allowed, problems may occur in combination with some programs. These restrictions do not apply to the separately stored full name.

UID and GID

The UID number is used for the internal identification of the user. In particular, the number is saved as additional information for each file so that it's clear who the file belongs to. There are rules for assigning UID numbers. First, root always has UID=0. UID numbers between 1 and 999 then are reserved for server services and daemons in most distributions. Accordingly, numbers starting from 1000 are provided for ordinary users. The GID number indicates to which group the user belongs. (Details on groups follow in Section 14.5.2.)

Home directory

The home directory is the place where users can store their private data. For ordinary users, the /home/<loginname> path is usually used for this purpose. The home directory also stores a user's personal configuration settings for various programs. For example, the .emacs file contains the configuration settings for the Emacs editor. Because the names of such configuration files usually start with a dot, they are invisible. They can be displayed using the ls -la command.

To ensure that new users immediately have sensible default settings for the most important programs, all files should be copied from /etc/skel to the newly created home directory when creating a new user. Many programs for creating new users perform this step automatically. The contents of /etc/skel thus represent the initial setting for each new user.

Shell

The *shell* is an interpreter that allows a user to execute commands after login. As there are several shells to choose from on Linux, the shell to be used must be specified in the passwd file. On Linux, this is mostly the bash shell. The passwd file must contain the full filename of the shell—for example, /bin/bash.

14.5.2 Group Management

The purpose of groups is to allow multiple users to share access to files. For this purpose, each user is assigned to a primary group (*initial group*). In

addition, a user can be assigned to any number of additional groups (*supplementary groups*); that is, the user can be a member of multiple groups.

The /etc/group file contains the list of all groups. The following lines show some group definitions in /etc/group. The following format applies:

Group name:Password:GID:Users list

The following lines are from the group file of an Ubuntu system:

```
root:x:0:
daemon:x:1:
bin:x:2:
sys:x:3:
adm:x:4:syslog,kofler
cdrom:x:24:kofler
...
kofler:x:1000:
huber:x:1001:
...
```

The assignment between user and group is done in two ways:

- The primary group of a user is stored in /etc/passwd. For user kofler, the primary group is also kofler; this is evident from the value 1000 in the GID column in the /etc/passwd file.
- Membership in additional groups is stored by specifying the user's name in the last column of the /etc/group file. For example, kofler also belongs to the adm and cdrom groups. This allows user kofler to perform admin tasks on Ubuntu systems and access CDs/DVDs.

For GID numbers, 0 is for root and 1 to 99 are for system services. GID=100 is usually reserved for the users group. GIDs greater than 1000 are then used for user-specific groups or may be defined for custom purposes.

Primary group

There are two common strategies for mapping between users and their primary groups:

- In the traditional method, which has been used on Unix/Linux for many years, all ordinary users are assigned to the primary users group. SUSE is still a follower of this very simple strategy today.
- Debian-based distributions as well as Red Hat and Fedora use a different method: each user gets their own primary group. In this case, there is a group of the same name for each of the users kofler and hoover. The users group no longer plays a role.

This method has advantages in certain circumstances—such as when several members of a secondary group create common files. However, these advantages only come into play with appropriate system administration. The differences are explained in detail on the Debian Wiki page at *https://wiki.debian.org/UserPrivateGroups*.

Assigning another secondary group

It's often necessary to assign an additional secondary group to a user. This allows you to give the user additional permissions, such as access to general-purpose input/output (GPIO) pins on the Raspberry Pi or permission to use `sudo`. Depending on the distribution, two commands are available to assign another group (here `admin`) to a user (here `bob`):

```
root# usermod -a -G admin bob  (all distributions, -a -G stands for append group)
root# adduser bob admin        (equivalent, Debian/Ubuntu only)
```

14.5.3 Passwords

Linux passwords usually consist of ASCII characters only. International special characters are basically allowed, but can easily cause problems, such as when typing on a keyboard that is not yet configured correctly. For security reasons, passwords should contain both uppercase and lowercase letters and at least one digit.

In Linux, passwords are stored as hash codes, which allow passwords to be checked but not reconstructed. The passwords can have any length. Current distributions use the hash algorithm SHA512, which is considered secure, in combination with a random initialization code for the password, the *salt*. This salt causes a different hash code to be stored for the same password each time. This makes it impossible to see in `/etc/shadow` whether multiple users have the same password. The algorithm is specified by the `ENCRYPT_METHOD` variable in `/etc/login.defs`.

To make life more difficult for potential attackers, the encrypted password codes are not stored directly in `/etc/passwd`, but in the separate `/etc/shadow` file. The advantage is that this file can be read only by `root`. In contrast, `/etc/passwd` and `/etc/group` are readable by all users of the system because they contain basic admin information. With `/etc/shadow`, on the other hand, it's sufficient if only the programs for password verification and change are allowed to access it. Potential attackers must therefore first gain `root` access before they can even read the encrypted password codes.

The following format applies to the `shadow` file:

Login:Password code:d1:d2:d3:d4:d5:d6:reserved

The following lines show an excerpt from a `shadow` file:

```
root:$y$Ecrkix...:14391:0:99999:7:::
daemon:*:14391:0:99999:7:::
bin:*:14391:0:99999:7:::
...
kofler:$y$TZR7...:14391:0:99999:7:::
```

For most system users, like `daemon` or `bin`, only an asterisk or exclamation mark is stored instead of a password. This means that there is no valid password, so logging in is impossible. The system accounts can still be used; programs that are first started with `root` privileges can later change ownership and then continue as `bin`, `daemon`, `lp`, and so on. This is exactly the case with most system processes: they are started by `root` during system startup and then change hands immediately for security reasons.

d1 through d6

The `d1` through `d6` fields can contain optional time information:

- `d1` indicates when the password was last changed (calculated as the number of days elapsed since January 1, 1970).
- `d2` specifies in how many days the password can be changed.
- `d3` specifies the maximum number of days until the password must be changed before it becomes invalid. (Details about the field can be found in `man 5 shadow`.)
- `d4` specifies how many days the user is warned before the password expires.
- `d5` specifies after how many days an expired account without a valid password will be completely deactivated.
- `d6` indicates how long it has been since an account has been deactivated.

Typically, default values are used for `d1` to `d3` so that the password can be changed at any time and remains valid indefinitely. `d1` through `d6`, however, can also be used to limit the validity of passwords, to automatically deactivate login accounts for a certain time, and so on—for example, to manage student accounts at a university or school.

chage

The `d1` through `d6` fields are set by the `chage` command (*change age*). In the following example, the user is forced to change their password immediately after the first login. Also, in the future, this user will have to change the password every 100 days. Finally, the account is only available until December 31, 2024. After that, any login will be blocked:

```
root# chage -d 0 -M 100 -E 2024-12-31 loginname
```

/etc/login.defs

A lot of parameters for the internal administration of passwords and logins are located in the `/etc/login.defs` file. The settings apply to `useradd`, `groupmod`, and so on. For example, `logins.def` specifies how many days passwords are valid by default, what range of values is used for new UIDs and GIDs, and so on.

The PAM configuration (see Section 14.6) also defines rules for password verification. The PAM rules are taken into account, among other things, by the `passwd` command and at every login.

Changing passwords

To change your own password, you need to run the `passwd` command. You will then first be asked for your old password and then prompted twice in a row to type in a new password. Only if both entries match will the new password be accepted. From then on you will have to use the new password every time you log in.

While ordinary users can only change their own password, `root` may also change the passwords of other users:

```
root# passwd hoover
New password: ******
Re-enter new password: ******
Password changed.
```

Password quality

In most distributions, PAM extensions ensure a minimum password quality. Often, passwords must be at least eight or nine characters long, differ from dictionary entries, contain numbers, and so on. The way in which this quality control is carried out depends on the distribution:

- In Fedora and RHEL, `pam_pwquality` ensures password quality. The configuration is done through `/etc/security/pwquality.conf`. A good introduction to how it works can be found at *https://deer-run.com/users/hal/linux_passwords_pam.html*.
- Arch Linux and SUSE use the `pam_cracklib` library instead. How to define rules that differ from the default settings is documented in detail at *https://deer-run.com/users/hal/linux_passwords_pam.html*.
- Debian and Ubuntu make use of the password tests integrated in the `pam_unix` PAM module. Details on how this works can be found via `man pam_unix`, where the description of the `obscure` parameter is particularly relevant. This parameter is active in Debian as well as in Ubuntu (`/etc/pam.d/common-passwd` file). If necessary, you can influence the minimum password length by the additional `minlen` parameter.

The password rules apply only to ordinary users. `root` may use the `passwd` command to choose arbitrarily bad passwords for itself as well as for any other user, although this is of course not recommended.

Generating Secure Passwords

If you frequently set up user accounts, it's convenient to use automatically generated passwords. The `mkpasswd` program from the Fedora or RHEL package `expect`, or `makepasswd` from the Debian or Ubuntu package of the same name, can provide help for this:

```
root# makepasswd
  aGjoQK1Ezo
```

To assign automatically generated passwords to new users in a script, it's best to use the `chpasswd` command.

Forgotten root password

What do you do if you forget your `root` password? In most distributions, you need a live or rescue system that you boot from a USB flash drive. Then, you must mount the `/dev/xxx` system partition of your Linux system in a directory of your choice. Using `chroot`, you can make this directory the new root directory. Then you can use `passwd` to reset the `root` password:

```
root# mkdir /rescue
root# mount -t auto /dev/xxx /rescue
root# chroot /rescue
root# passwd
root# <Ctrl>+<D>
root# reboot
```

If you want to prevent other users from changing the `root` password in the way just described, you must disable all boot media except the first hard disk in your computer's BIOS/EFI and password-protect the BIOS/EFI itself. But then you really must not forget this password! Even this method does not provide perfect protection as an attacker could easily remove your computer's hard disk/SSD and install it in their own computer. In that case, the procedure described previously works.

A much more effective way to secure a Linux system is to encrypt the entire system partition. The corresponding password must then be entered for each boot process, which is why this procedure is particularly useful for desktop installations. For server operation, encryption only makes sense if the server is easily accessible for you as an administrator! This is because even in this case you have to enter the password locally every time you restart. This is impossible for servers located in an external data center.

faillog

On Debian and Ubuntu, incorrect login attempts are logged in `/var/log/faillog`. Unlike many other logging files, this one uses a binary format. The configuration of the `faillog` file is done in `/etc/login.defs`.

You can use `faillog -u name` to determine how many incorrect login attempts have occurred for a particular user since the last valid login.

With `faillog -u name -m max`, a maximum number of failed login attempts for a given user can be fixed. If this number is exceeded, the login is blocked until `root` allows the login again by using the `faillog -u name -r` command. (This will reset the login counter.) You can also set the maximum login count in general via `faillog -m max`. However, you should then always also run `faillog -u root -m 0` so that `root` is excluded from this protection measure. Otherwise, you might not be able to log in even as `root` after another user has made several unsuccessful `root` login attempts.

faillock

On Fedora and RHEL, `faillog` is not available. A comparable function is provided by the `pam_faillock` PAM extension and the associated `faillock` command. These components are installed by default but are not active. You can find an example of a corresponding configuration at *https://unix.stackexchange.com/questions/182091*.

Group passwords

As with individual users, passwords can also be defined for groups (`gpasswd` command). But while passwords are strongly recommended for users, group passwords are uncommon. Their main disadvantage is that all group members must know the password, which complicates administration and makes the application inherently insecure. If group passwords are actually used, they are stored in the `/etc/gshadow` file. A group password must be entered when a user changes their currently active group using the `newgrp` command.

14.5.4 Interaction of the Configuration Files

The following lines summarize again how the `passwd`, `group`, and `shadow` files interact. For each user, `passwd` contains a line according to the following pattern:

```
# one line in /etc/passwd
kofler:x:1000:1000:Michael Kofler:/home/kofler:/bin/bash
```

Here, `kofler` is the login name; 1000 is both UID and GID; `Michael Kofler` is the full name; `/home/kofler` is his user directory; and `/bin/bash` is his shell. The UID must be a unique number because it is important for managing file access rights.

The corresponding line in `/etc/shadow` with the password information looks as follows:

```
# one line in /etc/shadow
kofler:$y$9dk0$...:13479:0:99999:7:::
```

The string after `kofler:` is the encrypted password. If the string is omitted, the login can be used without a password. If a `*` or `!` is entered instead of the string, the login is blocked.

The GID number in `/etc/passwd` must match a group from `/etc/group`. In many distributions, each ordinary user is assigned a group of the same name:

```
# one line from /etc/group
kofler:x:1000:
```

14.5.5 Network User Management

If you want to network several Linux computers and use NFS to enable mutual access to files, then you must make sure that the UID and GID numbers are consistent on all computers. This alone becomes quite time-consuming with several computers. If you now also want every user to be able to log in on every computer, and of course always under the same login name and with the same password, then you must constantly synchronize all `/etc/passwd` files. The administration effort then becomes huge.

To avoid this effort, a central server is usually used for user administration in such cases. There are many alternative methods and protocols available for authenticating clients, but describing them is beyond the scope of this book:

- Samba or Windows user management (Active Directory)
- Lightweight Directory Access Protocol (LDAP)
- Kerberos
- Network Information Service (NIS; obsolete)

14.6 PAM, NSS, and Nscd

After presenting the principles of user and group management in the previous section, here are a few more technical details. The following procedures are represented by the three abbreviations in the heading:

- **PAM**
  Pluggable authentication modules (PAMs) are central libraries that provide authentication functions to various commands.
- **NSS**
  The *name service switch* (NSS) decides which sources are used to resolve user, group, and host names.

- **Nscd**
  The *Name Service Caching Daemon* (nscd) is a cache for name queries. It can make the most recently required user, group, and host names available again particularly quickly.

14.6.1 PAM

A PAM is a library whose functions provide help with authentication tasks. When you perform a login or otherwise authenticate yourself or change your password on a Linux machine, the respective programs access the PAM library. Cron and PolicyKit also use PAM. You can use `ldd` to determine whether a particular command uses PAM libraries:

```
root# ldd /usr/bin/passwd | grep libpam
  libpam.so.0 => /usr/lib/libpam.so.0 (0x00007f7f98420000)
  libpam_misc.so.0 => /usr/lib/libpam_misc.so.0 (0x00007f7f9841b000)
```

The PAM documentation can be found on the project page at *https://github.com/linux-pam/linux-pam*.

PAM is configured by default to analyze the local password files, such as `/etc/shadow`. If another authentication method is also to be used (e.g., LDAP), the PAM configuration must be changed accordingly. Depending on the distribution, different tools can help you with this:

- **Fedora, RHEL**
  `authselect`
- **SUSE**
  YaST: **Security • User and Group Management**
- **Debian, Ubuntu**
  `pam-auth-update`

pam.conf

The configuration files are located in the `/etc/pam.d/` directory. In addition, the `/etc/pam.conf` file is analyzed as well. The configuration files contain rules line by line. Each rule consists of at least three parts or columns:

```
Type Response PAM module [module_arguments]
```

Entries in `pam.conf` must also be prefixed with the name of the service, such as `login` or `other` for standard entries. For the files in `/etc/pam.d`, the service name is derived from the filename.

Rule type (first column)

PAM distinguishes between four rule types. In Debian, SUSE, and Ubuntu, the `common-account`, *common-auth*, *common-password*, and *common-session* files contain default rules for these four types:

- `account`
  Allows limiting the services depending on the time of day, load, login location (e.g., console), and so on
- `auth`
  Concerns authorization (i.e., password query and verification) and the subsequent assignment of privileges, such as group memberships
- `password`
  Concerns the method for changing the password
- `session`
  Allows you to perform actions before or after the actual service: logging, mounting/unmounting file systems, showing the mailbox status, and so on

If the rule type is preceded by a minus sign, there will be no error message if a module specified in the rule is not available. This syntax variant allows the definition of rules that become active only when the corresponding PAM module is actually installed.

Response (second column)

The second column in the configuration files specifies how PAM should react when a rule is complied with or not. There are two ways to describe the response: either by a simple keyword (such as `required`, `requisite`) or by a value/result pair enclosed in square brackets (e.g., `[success=1 new_authtok_reqd=done default=ignore]`). The meaning of the four most important keywords of the simple syntax variant is explained in Table 14.4.

In the second syntax variant, you can specify multiple value/result pairs in the following format: `[value1=result1 value2=result2 ...]`. For `value`, there is a whole set of predefined keywords available that express the result of a rule. `result` can either be a number that specifies how many more rules should be skipped, or a keyword that specifies the desired PAM result (`ignore`, `bad`, `die`, `ok`, `done`, or `reset`). The following listing indicates how the keywords of the first syntax variant are expressed in the second notation:

```
requisite  = [success=ok new_authtok_reqd=ok ignore=ignore default=die]
required   = [success=ok new_authtok_reqd=ok ignore=ignore default=bad]
sufficient = [success=done new_authtok_reqd=done default=ignore]
optional   = [success=ok new_authtok_reqd=ok default=ignore]
```

Keyword	Response
`requisite`	In the event of a rule violation, the PAM function immediately returns a negative result and the other rules are no longer processed.

Table 14.4 Response to PAM Rule Violations

Keyword	Response
required	In case of a rule violation, the PAM function returns a negative result; further rules are processed, but their result is not taken into account.
sufficient	If the rule is followed, PAM immediately returns a positive result (unless there is already a violation of a previous requisite rule); no other rules are considered. In case of a rule violation, PAM continues with the next rule.
optional	The rule result is relevant only if it is the only rule for a particular rule type and service (such as su).

Table 14.4 Response to PAM Rule Violations (Cont.)

Module and options (third column)

The third column specifies the name of the PAM module that analyzes the rule. The behavior of the module can be influenced by options. Unfortunately, there is no central documentation of the permissible options and their meaning.

Standard configuration

The following listing, somewhat abbreviated for space reasons, summarizes Fedora's settings. Note that the default settings vary greatly depending on the distribution and are often spread across several files—for example, in common-xxx in Debian, SUSE, and Ubuntu:

```
# file /etc/pam.d/password-auth (Fedora)
# Generated by authselect. Do not modify this file manually.
auth       required                      pam_env.so
auth       required                      pam_faildelay.so delay=
2000000
auth       [default=1 ignore=ignore ...]   pam_usertype.so isregular
auth       [default=1 ignore=ignore ...]   pam_localuser.so
auth       sufficient                    pam_unix.so nullok try_
first_pass
auth       [default=1 ignore=ignore ...]   pam_usertype.so isregular
auth       sufficient                    pam_sss.so forward_pass
auth       required                      pam_deny.so

account    required                      pam_unix.so
account    sufficient                    pam_localuser.so
account    sufficient                    pam_usertype.so issystem
account    [default=bad success=ok ...]    pam_sss.so
account    required                      pam_permit.so

password   requisite                     pam_pwquality.so try_
first_pass  local_users_only
```

```
password   sufficient                    pam_unix.so yescrypt
shadow nullok  use_authtok
password   sufficient                    pam_sss.so use_authtok
password   required                      pam_deny.so

session    optional                      pam_keyinit.so revoke
session    required                      pam_limits.so
-session   optional                      pam_systemd.so
session    [success=1 default=ignore]    pam_succeed_if.so service
in crond  quiet use_uid
session    required                      pam_unix.so
session    optional                      pam_sss.so
```

PAM Configuration on Fedora and RHEL

On Fedora and RHEL, you should avoid modifying files directly in `/etc/pam.d`. The PAM configuration is generated by the `authselect` command, which has replaced the `authconfig` command commonly used in older distributions. Details about `authselect` can be found in the `man` documentation and at *https://fedoraproject.org/wiki/Changes/Authselect*.

14.6.2 Name Service Switch

When logging in, managing access rights to files, network access, and so on, all possible information about user and group names, UIDs and GIDs, host names, ports of network services, and so on is required. By default, this data is located in `/etc/passwd`, `/etc/group`, `/etc/hosts`, `/etc/services`, and other files.

In Unix or Linux terminology, access to this data is summarized under the term *name services*. The NSS, a collection of functions from the `libc` library, is responsible for this task. Similar to authentication, the data source for NSS can be set, for example, if an LDAP server provides the data. The crucial configuration file is `/etc/nsswitch.conf`. The following lines show the default configuration on Debian or Ubuntu:

```
# file /etc/nsswitch.conf (Ubuntu)
passwd:         files systemd sss
group:          files systemd sss
shadow:         files systemd sss
gshadow:        files systemd
hosts:          files mdns4_minimal [NOTFOUND=return] dns
networks:       files
```

```
protocols:      db files
...
```

The first column in `nsswitch.conf` designates the database or file. After the colon, keywords describe the method of accessing the data and optionally, in square brackets, the response to lookup results. The following list explains the most important keywords for the access methods. If a line names multiple access methods, they will be applied in sequence until one method succeeds. Further syntax details can be found in `man nsswitch.conf`:

- `files`
  Accesses the traditional configuration files (`/etc/passwd` and `/etc/group`).
- `compat`
  Same as `files`, but the user and group specifications can be preceded by the `+` and `-` signs. This increases the compatibility with NIS. `compat` cannot be combined with other keywords.
- `db`
  Reads the data from a BDB database file (BDB = *Berkeley Database*).
- `dns`
  Contacts a name server.
- `mdns`
  Uses multicast DNS (aka Zeroconf or Apple Bonjour/Rendezvous).
- `sss`
  Contacts the System Security Services Daemon (`sssd`). This is a local service that acts as a cache for external authentication services such as Kerberos or LDAP.
- `systemd`
  Determines users and groups temporarily created by systemd to run a service (`DynamicUser` option; see `man systemd.exec`).

The access methods require the corresponding library to be installed, such as `libnss_db` for `db`. If the library is missing, the corresponding keyword is simply ignored without reporting an error.

14.6.3 Name Service Caching Daemon

In SUSE, the `nscd` program is installed by default and activated during the boot of the computer. In most other distributions an optional installation is possible. If configured accordingly, the program remembers login, group, and host names as well as their IP numbers and makes them available to the local computer. It makes sense to use `nscd` especially if the user management is done by a network service (such as LDAP). `nscd` then serves as a

SUSE

cache for information and avoids unnecessary LDAP queries, which are often answered comparatively slowly.

/etc/nscd.conf

The configuration of `nscd` is done via `/etc/nscd.conf`. The file contains entries for several groups: `passwd` for login names, `group` for groups, `host` for host names, and so on. The settings for each group determine the duration for which the data is stored, the maximum number of entries to be managed, and so on.

14.6.4 System Security Services Daemon

Fedora and RHEL

Fedora and RHEL prefer `sssd` instead of `nscd`. `/etc/nsswitch` looks as follows:

```
# file /etc/nsswitch (Fedora, RHEL)
passwd:     files sss systemd
shadow:     files
group:      files sss systemd
...
```

`sssd` is not a complete replacement for `nscd`, but it provides other additional functions. In any case, using `nscd` and/or `sssd` is only appropriate if your Linux clients access a central authentication system.

14.7 Language Setting, Internationalization, and Unicode

This section is about two things:

- **Localization or language setting**
  This setting determines in which language error messages, menus, dialogs, help texts, and so on will be displayed. Also the formatting of the date, time, currency amounts, and the like changes accordingly.
- **Character set**
  The character set determines which codes are used to store letters. A *character set* describes the assignment between numeric codes and letters. Known character sets are ASCII (seven bits), ISO-Latin-n (eight bits), and Unicode (16 or 32 bits). Within Unicode, there are again different encodings. The most common one is UTF-8. ASCII characters are represented with only one byte. However, up to four bytes are needed to map rarer characters.

What Do l10n and i18n Mean?

In connection with the localization of programs, you will repeatedly come across the strange abbreviations *i18n* and *l10n*: these are the short forms

for *internationalization* (*i* plus 18 letters plus *n*) or *localization*, respectively. These abbreviations are also good search terms if you want to search the internet for more information.

Font

The font must not be confused with a character set. It's responsible for how a particular character is displayed on the screen. There are various fonts for this purpose (such as Arial, Courier, Helvetica, and Palatino, to name but a few well-known ones). Most fonts contain only a subset of all Unicode characters due to space limitations. If a character (such as an emoji) is not displayed correctly, this is usually because the character is missing from the font.

14.7.1 Setting the Localization and Character Set

Configuration tools

Depending on the distribution or desktop system, you can use different tools to configure the language. For all distributions, you must log in again for changed language settings to take effect. UTF-8 is almost always used as the character set.

- **Debian**
  `dpkg-reconfigure locales` or `localectl`
- **Fedora, RHEL**
  `localectl`
- **SUSE**
  YaST module **System • Language**
- **Ubuntu**
  Languages (`gnome-language-selector`)
- **GNOME**
  System settings **Region and Language** module
- **KDE**
  System settings **Regional Settings** module
- **Systemd**
  `localectl`

In addition, some display managers provide the option to select the desired language for the next session in the login dialog for the desktop system.

Configuration files

The configuration settings are stored in `/etc/default/locale` in almost all distributions. To set a new default language, you need to run the `localectl set-locale xxx` command:

```
root# localectl set-locale C.UTF-8          (neutral/English)
root# localectl set-locale en_US.UTF-8      (US English)
root# localectl set-locale fr_FR.UTF-8      (French)
root# localectl set-locale de_DE.UTF-8      (German)
```

A list of all possible settings is provided by `localectl list-locales`. If this list contains only very few entries for Debian-based distributions, you must first install or set up the corresponding languages. To do this, you need to call `dpkg-reconfigure locales`.

SUSE (still) moves away from the common Linux standards. The language setting is preferably done using YaST and is stored in `/etc/sysconfig/language`.

LC and LANG variables

Internally, both the localization and character set are controlled by environment variables such as `LC_CTYPE` and `LANG`. The `glibc` library, which is used in almost all Linux programs, is responsible for analyzing these variables. Localization can be performed category by category. This makes it possible, for example, to use the format common in Germany for dates and times, but still display error messages in English. Table 14.5 lists the most important variables.

Variable	Meaning
`LANG`	Determines the default value for all LC variables that have not been set
`LANGUAGE`	Allows the settings of multiple languages (see below)
`LC_CTYPE`	Determines the character set
`LC_COLLATE`	Determines the sorting order
`LC_MESSAGES`	Determines the display of messages, error messages, and so on
`LC_NUMERIC`	Determines the representation of numbers
`LC_TIME`	Determines the display of date and time
`LC_MONETARY`	Determines the representation of monetary amounts
`LC_PAPER`	Determines the paper size
`LC_ALL`	Overwrites all individual LC settings

Table 14.5 Important Localization Variables

Note that you can set all of these variables in `/etc/default/locale`, with one exception: `LC_ALL`! This variable can only be changed interactively or for testing purposes in `/etc/profile` (see *https://unix.stackexchange.com/questions/567826*).

Of course, not every program takes all categories into account; some programs even ignore the LC_ variables in their entirety. If individual categories are not set, programs use C or POSIX as the default value. This means that error messages appear in English, dates and times are displayed in American format, and so on.

LANG and LC_ALL

Instead of setting all variables individually, you can simply set the LANG variable, which uses the default LANG value for all undefined variables. The POSIX default setting remains only for LC_COLLATE. In most distributions, the entire language setting is done via the LANG variable.

LC_ALL has an even stronger effect than LANG. If this variable is set, all categories will have this setting, no matter how LANG or other LC_ variables are set.

LANGUAGE

A special case is LANGUAGE: this variable is not part of the glic variables. Rather, it is specific to the GNU gettext function. The function is used in many programs to output messages or error texts in the desired language. If LANGUAGE is defined, the variable for gettext has priority over all other variables (including LC_ALL). LANGUAGE allows you to set multiple languages. LANGUAGE=de:fr, for example, causes output to be preferentially performed in German. If no German translation is available, French will be used.

Localization files

For most programs, error messages and other texts for each language are located separately in their own directories, such as in /usr/share/locale/language/LC_MESSAGES. You can find more details about this at *https://www.gnu.org/software/libc/manual*.

Testing the localization

The easiest way to determine the current state of the localization setting is to use the locale command. This command also analyzes LANG and LC_ALL to determine the resulting settings. The following example shows the setting on my computer:

```
user$ locale
LANG=de_DE.UTF-8
LC_CTYPE="de_AT.UTF-8"
LC_TIME="de_AT.UTF-8"
...

LC_ALL=
```

To test the localization, you can execute any command with errors. The error message appears in the set language. If LANG is set to de_DE, the error message of the mount command looks as follows:

```
user$ mount /xy
mount: Konnte /xy nicht in /etc/fstab finden
```

English error messages, please!

If you work on a non-English system and want to research the background of an error on the internet, error messages in the local language are not very helpful. Whether you work in Germany, France, or Mexico, you want to see error messages in English!

To execute a single command with a different language setting without changing the entire configuration, you can simply prefix the command with `LANG=`. This means that the `LANG` variable will be reset, but only for the execution of the command! Alternatively, `LANGUAGE=en` works with a little more typing:

```
user$ LANG= mount /xy
mount: can't find /xy in /etc/fstab
user$ LANGUAGE=en mount /xy
mount: can't find /xy in /etc/fstab
```

To change the language for the entire course of a shell session, you can run `export`:

```
user$ export LANGUAGE=en
```

Permissible LC/LANG settings

You can determine a list of all possible settings by using `locale -a`. Usually the `x_y` notation is used, where `x` is replaced by two letters indicates the language and `y` by two letters indicates the country. You can use `en_US` (US English), `en_GB` (British English), `fr_FR` (French in France), or `de_DE` (German), for example. For the English default setting, the short notation `C` is allowed. Newer glibc versions also understand settings like `german` or `deutsch`.

The file `/usr/share/locale/locale.alias` contains a table that maps the allowed short notations to the full locale name.

Localization packages

Whether menus, dialogs, error messages, help texts, and so on are displayed in the correct language also depends on whether the necessary localization files are installed. For space reasons, this is often the case for only one or two languages used during the installation, such as for English and German. If you now want to use your distribution also in French, you need to install appropriate additional packages for GNOME, KDE, LibreOffice, Firefox, and so on. On SUSE and Ubuntu, the configuration tools listed in the introduction to this chapter will help you with this, but in other distributions, manual work is required in this case.

Not every Linux program has been localized for every language. There are particularly large gaps in online documentation—`man` pages, manuals, and help texts. If suitable localization files are missing, Linux displays English texts.

14.7.2 "Cannot Set/Change Locale" Error Message

Recently, I have been encountering the following error messages over and over again, especially during minimal installations in the cloud or in virtual machines:

```
Cannot set xxx to default locale: No such file or directory ...
 setlocale: LC_ALL: cannot change locale
```

This is apparently due to the efforts of many distributions to avoid installing more localization files than absolutely necessary. But if a `LANG` or `LC` variable then uses a localization that isn't installed, errors will occur.

The correct solution varies depending on the distribution. On Fedora and RHEL systems, the following command often helps, where you replace `XX` with the language you need (so, for example, `langpacks-de` for German):

```
root# dnf install -y langpacks-en langpacks-XX glibc-all-langpacks
```

locale-gen

On Debian and Ubuntu, `dpkg-reconfigure locales` is the fastest way to achieve this. Alternatively, you can search for the supposedly missing localization (such as `en_US.UTF-8`) in the `/etc/locale.gen` file) and remove the # character located at the beginning of the line. Then run `locale-gen`. This procedure recently helped me with an apparently faulty Manjaro installation in which `dpkg-reconfigure` was not available.

14.8 Hardware Reference

This book does not contain a separate chapter on hardware. Instead, the correct configuration of hardware components is covered in the corresponding chapters. Thus, if you have network problems, for example, Chapter 15 is the place to start.

This section has two tasks: On the one hand, it is intended to facilitate the search for further information on specific hardware components. On the other hand, you will find brief information on hardware topics that are neglected in the rest of the book. Of course, there are a lot of hardware components that are not described here or in other chapters of the book. That's because I don't have the testing capabilities that, say, a computer magazine has.

Research First, Then Buy!

Find out if your new hardware is Linux compatible *before you buy it*! Perform a search on the internet using the term *linux* and the hardware model name.

Device files Most hardware components are addressed via so-called devices—for example, `/dev/nvme0n1` for a modern SSD. The device files are dynamically created on demand by the `udev` system. For a list of the most important Linux device files, see Chapter 9, Section 9.9.

Kernel modules The drivers for countless hardware components are located in kernel modules. Some of these modules are loaded during system startup; the remaining modules are loaded only when needed. If the automatic loading of modules doesn't work, you should take a look at the `/etc/modprobe.conf` or `/etc/modprobe[.conf].d/*` file. The handling of modules and the function of these files are described in Chapter 21, Section 21.1.

The kernel messages show what happens when modules are loaded and whether the hardware can be initialized successfully. You can read these messages by using the `dmesg` command. `lspci -k` indicates which PCI device (i.e., the graphics card, the audio adapter, etc.) is served by which driver.

/proc and /sys files For many components, virtual file systems in the `/proc` and `/sys` directories provide detailed information. For an overview of such hardware files, see Chapter 21, Section 21.7.

Hardware overview To get an overview of the running hardware, you can execute the following commands: `lsblk`, `lscpu`, `lshw`, `lspci`, `lsscsi`, and `lsusb`. Once again, a look at the kernel messages with `dmesg` also provides useful information.

A great tool to determine a visually appealing summary of all hardware components and their associated drivers is the `inxi` command. The Perl script can be installed on most distributions without any problems. Without further options, it only outputs the key data of the computer. In the terminal, keywords of the output are highlighted in color, which makes the text more readable than in the following listings:

```
user$ inxi
CPU: 6-core Intel Core i7-8750H (-MT MCP-) speed/min/max: 1733/800/4100 MHz
Kernel: 6.3.1-arch1-1 x86_64 Up: 1h 6m Mem: 11689.0/31556.3 MiB (37.0%)
Storage: 2.29 TiB (32.5% used) Procs: 398 Shell: Zsh inxi: 3.3.27
```

With appropriate options, it displays detailed information about certain functions—for example, with `-A` for the audio system, `-G` for the graphics system, or `-E` for all Bluetooth components. `inxi -F` (think of *f* for *full*) provides all the data that the command can determine:

```
user$ inxi -A
Audio:
  Device-1: Intel Cannon Lake PCH cAVS driver: snd_hda_intel
  Device-2: NVIDIA GP107GL High Definition Audio driver: snd_hda_intel
```

```
  Device-3: C-Media USB Audio Device
    driver: hid-generic,snd-usb-audio,usbhid type: USB
  API: ALSA v: k6.3.1-arch1-1 status: kernel-api
  Server-1: PipeWire v: 0.3.70 status: active
```

14.8.1 CPU and Memory

CPU

The `/proc/cpuinfo` file shows which CPUs are running in your computer. The following heavily abbreviated output was created on a computer with an AMD Ryzen processor. Linux treats the cores like independent processors. The `cpu MHz` line contains the maximum base frequency, which can be exceeded for a short time in many models (depending on the manufacturer, this process is referred to as *turbo boost*, *turbo core*, etc.):

```
user$ cat /proc/cpuinfo
processor       : 0
model name      : AMD Ryzen 5 3600 6-Core Processor
cpu MHz         : 2195.694
...
processor       : 1
model name      : AMD Ryzen 5 3600 6-Core Processor
...
```

Section 14.9 contains some tips on how you can influence the behavior of the CPU.

Memory (RAM)

Information about the available memory can be obtained using the `free` command. `lsmem` shows the internal organization of the memory.

If you suspect your computer to have defective memory modules, the *MemTest86* program is a good way to test the RAM. In some distributions (such as Fedora), the program can be started conveniently in the GRUB menu after switching on the computer. If that doesn't work for you, you can find an ISO image at *https://memtest86.com*. Transfer this image to a USB flash drive and use it to restart the computer.

14.8.2 Power Management

ACPI

Advanced Configuration and Power Interface (ACPI) controls the power-management functions of all commercially available PCs and notebook computers. ACPI is supported by Linux, which you can check by using `dmesg | grep -i acpi`.

Suspend

ACPI supports various sleep modes in which the computer consumes little (standby, suspend mode) or no (hibernate mode) power at all. In most distributions or desktop systems, you put the computer into sleep mode via the system menu.

In standby mode, the CPU is put into a sleep mode in which it requires little power. The RAM continues to be supplied with power, but displays and data storage media are switched off.

In hibernation mode, on the other hand, the current RAM content is saved in the swap partition of the hard drive or SSD and the computer is then completely switched off. In this state, it no longer needs any power at all. This requires the swap partition or file to be sufficiently large! When waking up, the memory is read from the hard drive or SSD again.

In addition, all hardware components must be reinitialized. This process is very complex and requires an optimal interaction of the Linux kernel, its modules, and the ACPI system.

Waking Up from the Calm...

Unfortunately, my personal experiences with hibernation mode are mostly negative. Unless an error had already occurred when trying to activate it, it was often not possible to wake up the computer from sleep mode again. For this reason, many distributions have moved away from offering the activation of hibernation mode in the system menu at all.

Things look much better with standby mode, which often works without any problem in simple cases. *Simple* here means that no external devices are connected to the computer via USB-C hubs and no brand-new hardware components are in use. Otherwise, it can happen that the computer wakes up again, but the screen remains black or the speakers remain mute.

Saving Energy, Protecting the Battery

A collection of tips on how to minimize your notebook's power consumption and possibly optimize your battery life follows in Section 14.10. There, I will introduce the `powertop` and `tlp` programs, among others.

14.8.3 Interfaces and Bus Systems

Serial and parallel interfaces

On Linux, serial or parallel interfaces are accessible via the `/dev/ttyS<n>` or `/dev/lp<n>` device files. You are most likely to encounter these interfaces, which are no longer common in the PC sector, in min-computers or embedded devices.

SATA, SCSI

Hard disks, DVD drives, and other data storage media are usually connected to the computer via the SATA or SCSI bus systems (see Chapter 18, Section 18.3). Particularly fast SSDs use the PCI Express (PCIe) standard instead. Information about the state of such devices is provided by the `lsscsi` command from the package of the same name and the following files:

*/sys/bus/scsi/**
*/proc/scsi/**

USB

Universal Serial Bus (USB) is used to connect the computer to various external devices—from a mouse to an external hard drive. The required USB kernel modules are loaded automatically.

The `/dev/bus/usb` and `/sys/bus/usb` directories contain information about all connected USB devices. A detailed listing of all USB interfaces and devices is provided by `lsusb -v` (`usbutils` package). In case of USB problems, you can use `usbreset` to perform a reset during operation. (Disconnect all external data media beforehand!)

PCI

The best way to find information about PCI components in your computer is to use the `lspci` command. The files in `/proc/bus/pci/` and `/sys/bus/pci/` contain the same information but are much more difficult to interpret. The following output is heavily abridged for reasons of space:

```
root# lspci
00:00.0 Host bridge: Intel Corporation 8th Gen Core Processor Host
Bridge/DRAM Registers (rev 07)
00:02.0 VGA compatible controller: Intel Corporation CoffeeLake-H GT2
[UHD Graphics 630]
00:1f.6 Ethernet controller: Intel Corporation Ethernet Connection
(7) I219-V
02:00.0 Non-Volatile memory controller: Samsung Electronics Co Ltd
NVMe SSD Controller SM981/PM981
...
```

Graphics and network functions

Information on configuring LAN and Wi-Fi interfaces follows in Chapter 15. The use of graphics cards or GPUs requires separate drivers. Their configuration is the subject of Chapter 17.

14.8.4 Bluetooth

Bluetooth is a method for hardware devices to communicate via radio. Bluetooth is mainly used in small electronic devices (keyboards, mice, speakers, smartphones, and so on). In desktop systems such as GNOME or KDE, convenient interfaces help you set up Bluetooth devices. However,

patience is sometimes required as it often takes some time before the devices are recognized.

Behind the scenes, the implementation of the Bluetooth stack is the responsibility of the `bluetoothd` daemon in interaction with `udev` for the management of Bluetooth devices. The corresponding configuration files are located in the `/etc/bluetooth` directory. More information about Bluetooth on Linux can be found at *http://www.bluez.org*.

bluetoothctl

Only in rare cases is it necessary to configure Bluetooth at the command line level. One possible scenario is to use the Raspberry Pi without a monitor if control and communication is to be done exclusively via SSH. In such cases, it's best to use the `bluetoothctl` command, which is intended for interactive use. After it starts, you enter a command mode where you can execute various commands: `list` lists all Bluetooth controllers (such as a USB adapter), and `devices` lists all devices in radio range. Using `exit` or Ctrl+D, you can return to the input mode of the terminal.

To connect a new Bluetooth device to your computer, you should do the following within a `bluetoothctl` session:

- Activate the coupling mode using `pairable on`.
- Activate the scan mode using `scan on`. The program then lists all detected devices within radio range. This process can take quite some time as individual devices are listed again and again. When you have found the device you want, you can simply switch the mode off again using `scan off`.
- Using `agent on` lets you activate a so-called Bluetooth agent. It takes care of the authorization of new devices (for keyboards: password entry).
- Then use `pair xx:xx:xx` to initiate the connection setup to a device. If you use a keyboard, you will then be prompted to type in a six-digit code. Do not forget to finish the input by pressing Enter! You can recognize a successful pairing by the *pairing successful* message. For devices without input options (keyboard, speakers, etc.), pairing is fortunately possible without entering a code.
- And by using `trust xx:xx:xx`, you make clear to the Bluetooth system that you trust the device.
- Using `connect xx:xx:xx`, you indicate that you actually want to use the device. (`bluetoothctl` could have realized that by now!) If everything works out, the response is *connection successful*. The device can now be used!
- `info xx:xx:xx` displays the connection status and various other information about the device.

It's important to keep pressing the Bluetooth button (pairing button) on Bluetooth devices that you want to reconfigure so that the device signals to the environment that it's ready to connect. On many devices, a blue LED then flashes. If there is no such button, you can also try turning the device off and on again.

The following lines show in slightly abbreviated form the inputs and outputs for configuring a Bluetooth keyboard. All inputs and outputs take place in a terminal window:

```
user$ bluetoothctl
[bluetooth]# agent on
[bluetooth]# pairable on
[bluetooth]# scan on
Discovery started
[CHG] Controller 00:1A:7D:DA:71:13 Discovering: yes
[NEW] Device 70:10:00:1A:92:20 70-10-00-1A-92-20
[CHG] Device 70:10:00:1A:92:20 Name: Bluetooth 3.0 Keyboard
...
[bluetooth]# scan off
[bluetooth]# pair 70:10:00:1A:92:20
Attempting to pair with 70:10:00:1A:92:20
[CHG] Device 70:10:00:1A:92:20 Connected: yes
[agent] PIN code: 963064
[CHG] Device 70:10:00:1A:92:20 Paired: yes
Pairing successful
[bluetooth]# trust 70:10:00:1A:92:20
[bluetooth]# connect 70:10:00:1A:92:20
[bluetooth]# info 70:10:00:1A:92:20
Device 70:10:00:1A:92:20
        Name: Bluetooth 3.0 Keyboard
        Paired: yes
        Trusted: yes
        Connected: yes
    ...
[bluetooth]# exit
```

The Bluetooth configuration for a particular device is stored in `/var/lib/bluetooth/id1/id2/info`. Here, `id1` is the ID code of the Bluetooth controller and `id2` is the ID code of the Bluetooth device.

Forcing the connection

The Bluetooth device should now also be detected correctly at the next startup. In my tests on a Raspberry Pi, however, I failed at this point with some Bluetooth devices. The solution was to execute the `connect` statement at the end of the boot process:

```
# file /etc/rc.local
...
echo -e "connect FC:58:FA:A0:4D:E7\nquit" | bluetoothctl
exit 0
```

14.8.5 Hotplug System

USB flash drives and hard disks, monitors, loudspeaker boxes, and so on can be connected or removed during operation. Linux must respond to the changed hardware situation quickly and as automatically as possible. This task is performed by the hotplug system:

- **Kernel**
  The kernel detects changes to the hardware—for example, that the user has plugged in a USB flash drive.
- `udev`
  The kernel creates new device files via `udev` (see Chapter 9, Section 9.9) and starts appropriate programs to manage the new devices or to send notifications to the desktop system. Rule files from the `/lib/udev/rules.d` and `/etc/udev/rules.d` directories are analyzed in this process.
- **DeviceKit**
  DeviceKit helps manage some devices or components. It consists of the `libudisk2` and `libgudev` libraries, which are usually packaged in packages of the same name. For hard disks (partitions) and external disks, the programs and scripts of the `udisks2` package are responsible. Power management is taken care of by the rules and programs of the `upower` package. You can find information on this topic at the following web pages:
  - *https://freedesktop.org/wiki/Software/DeviceKit*
  - *https://freedesktop.org/wiki/Software/udisks*
  - *https://upower.freedesktop.org*
- **Desktop**
  In KDE, the *Solid* framework is responsible for processing D-Bus messages. On GNOME, Nautilus takes care of the external disks in interaction with PolicyKit (see Chapter 10, Section 10.4). The configuration is done in the **Removable Media** module in the system settings.
- **D-Bus**
  The D-Bus communication system is used for the communication between the different levels of the hotplug system. Based on the `libdbus` library, communication can be done directly between two programs. If messages are to be exchanged between multiple programs, the `dbus-daemon` background program is used as the central exchange.

D-Bus is implemented outside the kernel. Plans to integrate the communication system into the kernel as part of the `kdbus` project have failed.

Thunderbolt and USB-C

The Thunderbolt protocol makes it possible to route the PCIe bus to the outside and connect various external devices to a computer—from high-resolution monitors to network adapters. The first Thunderbolt implementations were still based on DisplayPort; USB-C is now more common. Thunderbolt has been part of the USB standard since version 4.

Basically, Thunderbolt works without problems on Linux. However, Thunderbolt can be a security risk, among other things because the PCIe bus gives direct access to the RAM. For this reason, a notebook computer's Thunderbolt outputs can (and should) be protected in the BIOS/EFI. They are then not available until the user explicitly releases them during operation. This, in turn, is associated with disadvantages; for example, a keyboard connected to the USB-C hub cannot be used to enter the hard drive encryption password during the boot process.

On GNOME, security-critical devices connected via Thunderbolt can be enabled with RAM access in the **Privacy • Thunderbolt** module of the system settings. (Ordinary USB devices and monitors work without explicit permission, which is why the Thunderbolt module in Figure 14.2 somewhat misleadingly claims that no devices are connected at all.)

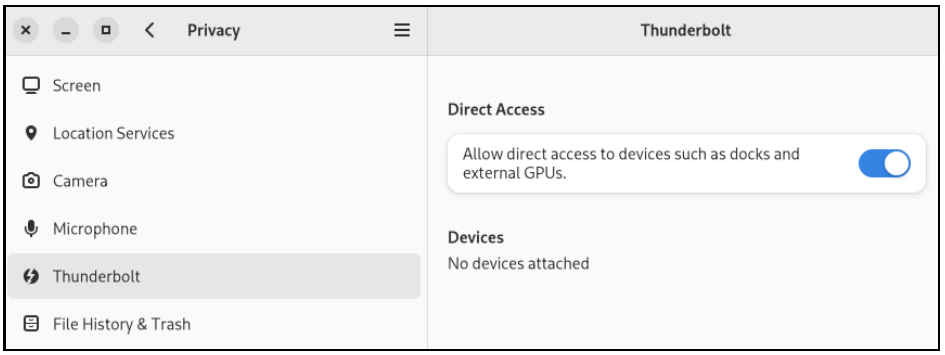


Figure 14.2 Thunderbolt Configuration in GNOME

At the command level, you can use `boltctl` to list detected Thunderbolt devices (`boltctl list -a`) and authorize them if necessary (`boltctl authorize`). You can read more details about the safe use of Thunderbolt devices in Christian Kellner's excellent blog, as well as a couple of other websites:

- *https://christian.kellner.me*
- *https://juho.tykkala.fi/Lenovo-Thunderbolt-3-dock-Linux*
- *https://fedoraproject.org/wiki/Changes/ThunderboltEnablement*

14.8.6 Audio System

ALSA

Advanced Linux Sound Architecture (ALSA) is responsible for the control of sound cards at the lowest level in the kernel. For audio controllers supported by the kernel, the required ALSA module is loaded automatically. The names of all ALSA modules start with `snd`. The `lsmod | grep snd` command therefore provides a quick overview of all active ALSA modules. Access to various sound functions is provided by files in the `/proc/asound` directory.

You can configure the ALSA system by using the `/etc/alsa/*`, `/etc/asound.conf`, and `.asoundrc` files. When the computer is shut down or rebooted, the init system saves or restores the volume settings.

For ordinary use of the audio system, there is no need to change the ALSA configuration. The hardware detection should succeed automatically. If you have special audio requirements (musicians), want to differentiate between several audio cards, or have other special requests, you'll find comprehensive background information about ALSA and its configuration at the following website and the associated wiki: *https://alsa-project.org*.

ALSA tools

For the direct use of ALSA, various commands are available (`alsa-utils` package), the most important of which are briefly introduced here: `alsactl` saves or loads all ALSA settings, such as the last set volume; `alsamixer` changes the volume or the input level of various ALSA audio channels (see Figure 14.3); `aplay` plays an audio file; and `arecord` records an audio file.

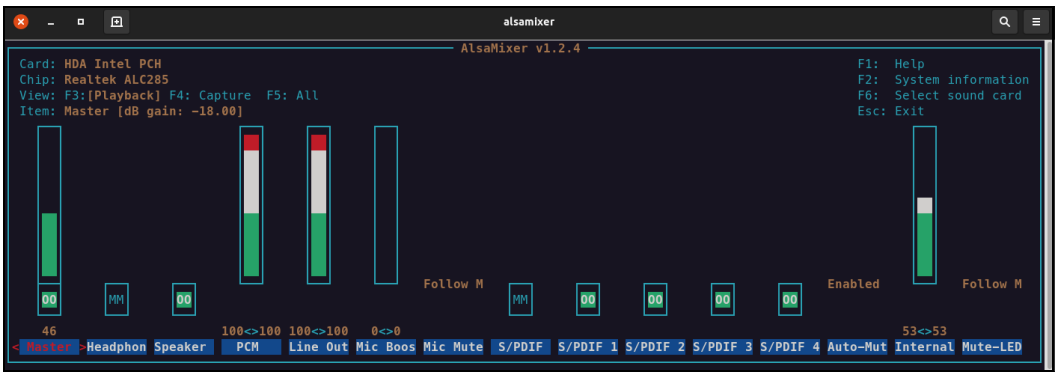


Figure 14.3 ALSA Configuration in Terminal with alsamixer

Troubleshooting the Audio System

If the speakers remain silent, the cause is often just a volume control set to 0. Modern distributions sometimes lack graphical user interfaces to set all audio inputs and outputs individually. A solution to this problem is as

follows: Start `alsamixer` in a text console. Now you can use the cursor keys to select the channels and adjust their levels. `M` switches a channel completely on or off again (*mute*).

The `speaker-test` ALSA command is helpful for initial tests of the audio system. When executed with the following parameters, the words *front left* and *front right* should be heard alternately in the left and right speakers:

```
speaker-test -t wav -c2
```

Audio libraries

The topic of audio often seems more trivial than it actually is. In the past, it was enough to address the (only) speaker output with an MP3 player. Today, multiple signal sources are the norm, from YouTube videoes in the web browser to an audio player to the notebook's internal microphone and an external one for video conferencing. Things are similarly complex on the output side: the built-in speaker is joined by a headset, the monitor connected via HDMI with speakers, and a Bluetooth box. The user wants to be able to easily change the input and output source at any time. Ideally, Linux should restore all settings at the next reboot.

For this reason, most audio programs do not use ALSA directly, but rely on audio frameworks. This intermediate layer between the low-level system ALSA and the actual audio applications simplifies programming, makes audio applications network-compatible, and ensures conflict-free cooperation of simultaneously running audio programs.

The problem now is that there is currently no unified audio architecture above ALSA: KDE and GNOME each go their own way. Sophisticated audio applications, for which the existing audio libraries are insufficient, reimplement elementary audio functions themselves. It is therefore difficult to develop audio programs that simply work regardless of the desktop system.

The following points briefly introduce some common audio systems:

- **GStreamer**
  The GStreamer library is a comprehensive multimedia framework used by many GNOME programs. Thanks to its plugin architecture, it can be extended well. (Most codecs are implemented as plugins.) The GStreamer library doesn't contain its own sound daemon; merging multiple audio signals is handled directly by ALSA. You can find information on this topic at *https://gstreamer.freedesktop.org*.
- **Phonon**
  The multimedia foundation of KDE is called Phonon. The library provides a uniform programming interface for the use of audio and video

functions, which uses existing multimedia libraries (often GStreamer or VLC; occasionally also Xine). Phonon is also used by the Qt library as a multimedia interface.

- **PulseAudio**
  PulseAudio is a network-enabled sound server that has been used in almost all Linux distributions for several years. Today, however, PulseAudio is more and more often replaced by PipeWire (see the following point).

  PulseAudio enables you to control audio streams and assign them to different audio cards or output devices by using the `pavucontrol` program. PulseAudio automatically detects external audio hardware (such as USB speakers).

- **PipeWire**
  The latest star in the Linux audio firmament is *PipeWire*. This is a new audio and video server. Thanks to a compatibility layer for PulseAudio, the changeover, which has already been completed by many distributions, has gone unnoticed by most users.

  One advantage of PipeWire is its lower latencies. At the same time, PipeWire enables secure screen sharing and remote desktop applications on Wayland. For this reason alone, there will probably be no way around PipeWire in the future. A good description of the concepts of PipeWire can be found at *https://venam.nixers.net/blog/unix/2021/06/23/pipewire-under-the-hood.html*.

  Various `pw` commands (in Fedora in the `pipewire-utils` package) help you to control and analyze the audio system. For example, `pw-top` lists all active audio components, including latency times.

Distributions for musicians

For musicians or professional audio users, there are separate distributions that are specially optimized for the trouble-free use of audio programs and often use a Linux kernel optimized for real-time operation. Popular distributions include, for example, Ubuntu Studio (*https://ubuntustudio.org*) and AV Linux (*http://www.bandshed.net/avlinux*).

14.9 CPU Tuning

Modern CPUs provide many options to analyze and influence their behavior. In this section, I'll explain how you can determine the current clock frequency and temperature and how to disable the turbo boost function. The examples refer to Intel CPUs. AMD CPUs provide quite similar control options but require different tools and use different `/proc` or `/sys` paths.

Collecting information on the CPU

You can get information about the CPU in your computer using the `lscpu` command or via `cat /proc/cpuinfo`. The easiest way to determine the number of CPU cores is to use the `nproc` command. If the CPU supports hyperthreading, then `nproc` displays twice as many cores as there actually are. To find out how many "real" cores are available, you can use the second of the following commands, which extracts this information from `/proc/cpuinfo`:

```
user$ nproc
12
user$ grep ^cpu\\scores /proc/cpuinfo | uniq | awk '{print $4}'
6
```

14.9.1 Controlling the CPU Frequency

Determining the CPU frequency

The CPU typically controls at which clock frequency it runs by itself. If there is little to do at the moment, the clock frequency is reduced. Conversely, the clock frequency can be increased for some time in the so-called turbo boost mode to complete complex tasks faster. However, this increases the CPU temperature considerably, which is why the amount of time for which the CPU can run at maximum clock frequency is strongly limited by the cooling. This is especially true for notebook computers.

The following command provides the current clock frequency of all real and virtual cores:

```
user$ cat /sys/devices/system/cpu/cpu*/cpufreq/scaling_cur_freq
1000033
1658625
1519742
1198860
...
```

You can get even more detailed information using the `cpupower` command, which can be found in the `linux-tools-common` package on Debian and Ubuntu:

```
root# apt install linux-tools-common linux-tools-generic

user$ cpupower frequency-info
analyzing CPU 0:
  ...
  hardware limits: 800 MHz - 4.10 GHz
  available cpufreq governors: performance powersave
```

```
  current policy: frequency should be within 800 MHz and 2.20 GHz.
                  The governor "powersave" may decide which speed to
                  use within this range.
  current CPU frequency: 1.00 GHz (asserted by call to kernel)
  boost state support: Supported: yes, Active: yes
```

Blocking the turbo boost

If you want your notebook computer to run as long and quietly as possible, it's recommended to deactivate the turbo boost mode:

```
user$ cat /sys/devices/system/cpu/intel_pstate/no_turbo          (current state)
0
root# echo 1 > /sys/devices/system/cpu/intel_pstate/no_turbo    (turbo boost off)
root# echo 0 > /sys/devices/system/cpu/intel_pstate/no_turbo    (turbo boost on)
```

Setting the CPU governor

The so-called governor of the CPU is responsible for regulating the clock frequency. Current Intel CPUs only support the `powersave` and `performance` modes. In `powersave` mode, the clock frequency of all cores is controlled dynamically and as energy-efficiently as possible (default state). In `performance` mode, all cores usually run at the same clock frequency, although higher frequencies are also permitted at low loads. The following commands show how you can determine or change the current governor mode:

```
user$ cat /sys/devices/system/cpu/cpu*/cpufreq/scaling_governor
root# for cpu in /sys/devices/system/cpu/cpu*/ \
                  cpufreq/scaling_governor; do
        echo powersave > $cpu
      done
root# for cpu in /sys/devices/system/cpu/cpu*/ \
                  cpufreq/scaling_governor; do
        echo performance > $cpu
      done
```

Even more configuration options

With a little effort, you can influence further parameters of the CPU and, for example, set the minimum or the maximum allowed clock frequency, how many cores can run at maximum at the same time, under which circumstances the clock frequency should be changed, and so on. Some of these tricks are documented on the following pages:

- *https://vstinner.github.io/intel-cpus.html*
- *https://wiki.archlinux.org/title/CPU_frequency_scaling*
- *https://www.kernel.org/doc/Documentation/cpu-freq/governors.txt*

Graphics tools

Depending on the distribution or desktop system, you can set the CPU power directly in the graphical user interface. In current GNOME versions,

you switch between three power modes in the **Power** module of the system settings or in the **Quick Settings** menu at the top right of the panel. On Pop!_OS, you can also use the `system76-power` command or the corresponding GNOME extension.

Behind the scenes, `power-profiles-daemon` is mostly used as the backend. This background service cannot be used simultaneously with other programs that also control the CPU (such as TLP).

14.9.2 Monitoring the CPU Temperature

If you want to know the current temperature of your computer's CPU, you need to install the `lm-sensors` or `lm_sensors` package. After installation, you can run the `sensors-detect` command as `root`. This command determines which hardware components provide information about their state. Besides the CPU, this can also include the hard drive, the graphics card, or various fans that report their speed. Finally, the required kernel modules are stored in the `/etc/modules` configuration file (Debian/Ubuntu) or in `/etc/sysconfig/lm_sensors` after a query.

After this preparatory work, the `sensors` command provides an up-to-date temperature list:

```
root# sensors
...
coretemp-isa-0000
Package id 0:  +53.0°C  (high = +100.0°C, crit = +100.0°C)
Core 0:        +50.0°C  (high = +100.0°C, crit = +100.0°C)
Core 1:        +51.0°C  (high = +100.0°C, crit = +100.0°C)
...
pch_cannonlake-virtual-0
Adapter: Virtual device
temp1:        +44.0°C
```

There are various programs that display this data more elegantly. For first experiments, you can use `xsensors`, which shows the data of all sensors in one window. On Ubuntu, you can install the `psensor` package (see Figure 14.4).

The *https://extensions.gnome.org* website also provides various GNOME extensions that display the temperature and other performance data in the panel. Unfortunately, these extensions are rarely maintained in the long term and often stop working with the next GNOME update.

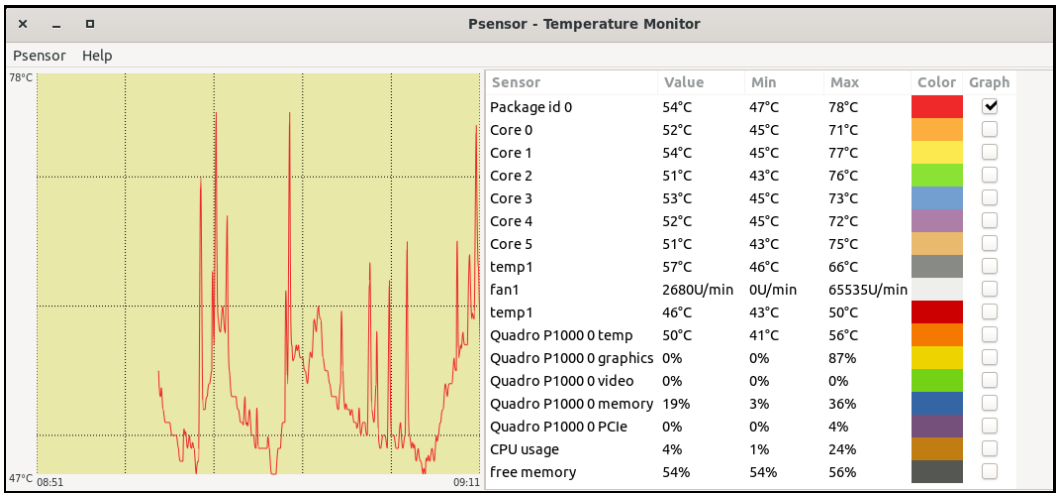


Figure 14.4 Temperature Display Using psensor

14.10 Notebook Optimization

Annoyingly, many notebook computers run longer in battery mode on Windows than on Linux. This is mainly due to better optimized drivers. This section provides some tips on how you can optimize both the runtime and the life of your notebook battery on Linux. The section also briefly covers the topic of fan control: you can influence the fan behavior on Linux for selected models.

CPU and Graphics Tuning

To make your notebook computer run as long as possible, you should deactivate the turbo boost mode of your CPU and activate the `powersave` governor. If you have a notebook with NVIDIA hybrid graphics, you should only use the Intel GPU if possible. For tips on this, see Chapter 17, Section 17.3.

14.10.1 powertop

The `powertop` command from the package of the same name helps you to search for programs that cause high power consumption or torpedo the power-saving functions of the CPU. The program provides tips on how you can minimize energy consumption. In the following command, `LANG=` causes `powertop` to display menus and status texts. This is especially useful if you want to search the internet for details about individual parameters:

```
root# LANG= powertop
```

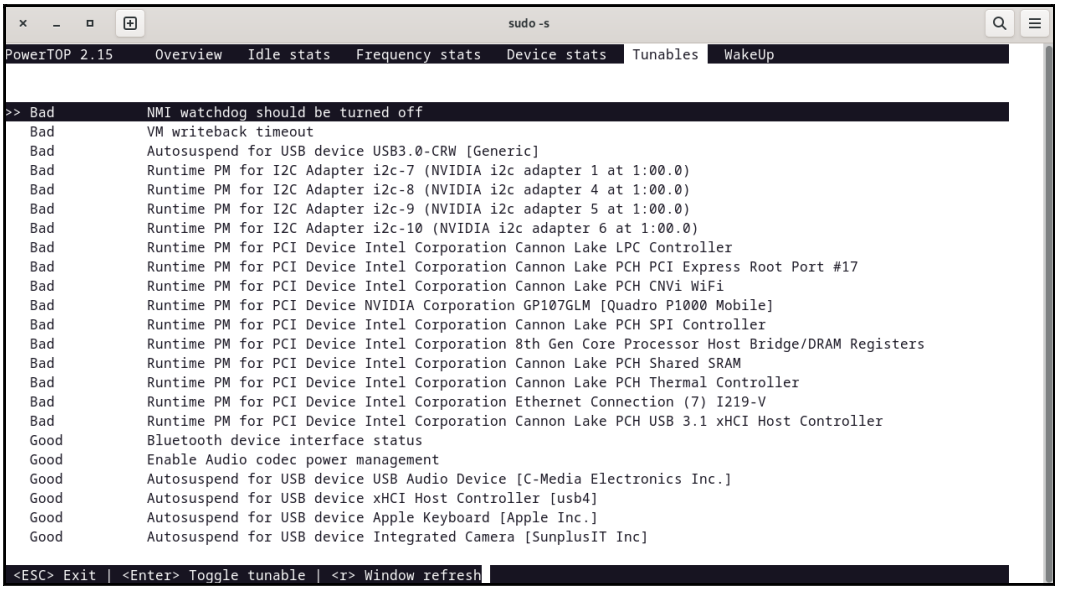


Figure 14.5 The Tunables View of powertop

After it starts, you can switch between several status pages by pressing the Tab key. These pages show which processes wake the CPU from an idle state and how often, how often the CPU is in what idle state, how often the CPU runs at what clock frequency level, which devices are used and how much, and so on.

The most interesting page from the point of view of energy-saving functions is called **Tunables** (see Figure 14.5). There, `powertop` shows a list of settings whose current state can be **Bad** or **Good**. You can then select individual points with the cursor keys and change them by pressing Enter. `powertop` shows which command it is executing.

Then you can try changing individual settings step by step and test what effect this has. Does the energy consumption of the previously charged notebook, recorded with a power meter, actually drop noticeably? Do the disabled functions cause problems? Can USB devices still be used? Does switching the Wi-Fi adapter on and off still work? Does the audio system work without noise? And so on.

The changes made using `powertop` are only valid until the next computer restart. To activate the power-saving measures permanently, you must enter the commands displayed by `powertop` (such as `echo '1' > /sys/xxx`) into a file that is executed at every system start. In some distributions, `/etc/rc.d/rc.local` is suitable for this purpose. If necessary, you must create this file and mark it as executable by using `chmod a+x`.

A radical solution would be to enter `powertop --auto-tune` in `rc.local` instead of individual tuning commands. Then `powertop` simply performs all known optimization measures. Unfortunately, `powertop` often overshoots the mark with this: What good is it if the notebook runs for an hour longer than before, but the network connection only works unreliably, or the connection to the Bluetooth mouse is constantly interrupted?

Note that `powertop --auto-tune` will not work until `powertop` has run on battery power for about half an hour beforehand and has gotten a picture of the activity on your notebook.

Creating a list of all tuning commands

You can get a list of all possible tuning commands by running `powertop` with the `--html` option. After a measurement time of about 20 seconds, the command then generates the `powertop.html` file, which contains various statistical data as well as a summary of all tuning parameters. For more tips on using `powertop`, see the Arch Linux wiki at *https://wiki.archlinux.org/title/powertop*.

14.10.2 TLP

TLP is a background service that implements various power-saving measures for Linux. TLP can be configured with countless options, but thanks to its sensible default values, much is already gained by simply installing TLP. The service is basically suitable for all notebook computers. However, some features are only available for Lenovo ThinkPads, including the battery-charging control described in the following section.

You may be wondering what TLP stands for. The service's FAQs reveal that TLP is not an acronym, but simply a name (see *https://linrunner.de/tlp/faq/misc.html*).

powertop versus TLP

`powertop` and TLP have a similar task. `powertop` is ideal for analyzing power consumption, while TLP is optimized to achieve reasonable basic settings with minimal configuration effort. However, it is not expedient to use both programs in parallel—that is, to run TLP *and* `powertop --auto-tune`—at system startup.

Installation

Most common distributions provide TLP packages. On Ubuntu, if you want to make sure you get the latest TLP version, it's best to set up an additional package source:

```
user$ sudo add-apt-repository ppa:linrunner/tlp
user$ sudo apt install tlp tlp-rdw
user$ sudo apt install tp-smapi-dkms acpi-call-dkms (for Lenovo Thinkpads only)
```

In the future, TLP will be executed automatically at computer startup as well as every time the computer switches between battery operation and power supply. The program then makes some settings and terminates again, so it does not run as a background service. To activate TLP after the installation without rebooting, you should run this command:

```
user$ sudo tlp start
```

To obtain TLP status information, you can use the `tlp-stat` command:

```
user$ tlp-stat -s
System          = LENOVO ThinkPad P1 20MD0001GE
BIOS            = N2EET56W (1.38 )
OS Release      = Arch Linux
Kernel          = 6.2.11-arch1-1
/proc/cmdline   = initrd=initramfs-linux.img cryptdevice=UUID=44fb...

Init system     = systemd
Boot mode       = UEFI

+++ TLP Status
State           = enabled
RDW state       = not installed
Last run        = 15:32:40,     21 sec(s) ago
Mode            = AC
Power source    = AC
```

Configuration

The `/etc/default/tlp` or `/etc/tlp.conf` file contains the TLP configuration along with brief explanations of all parameters. For example, the following options enable the `powersave` governor in battery mode, while the `performance` governor is to be used as soon as a power supply is available:

```
# file /etc/default/tlp oder /etc/tlp.conf
...
CPU_SCALING_GOVERNOR_ON_AC=performance
CPU_SCALING_GOVERNOR_ON_BAT=powersave
```

You can find numerous TLP configuration tips on the following page: *https://linrunner.de/en/tlp/tlp.html*.

14.10.3 Controlling the Battery-Charging Behavior

If your notebook is connected to an AC adapter most of the time, then the battery is fully charged all the time. This reduces the service life of the battery. It would be better if the battery is only charged to approximately 40 to 60% power in such phases. There are notebooks that provide corresponding settings in the BIOS/EFI (such as **FlexiCharger** in Tuxedo notebooks).

On some Lenovo notebook models, TLP lets you control under what circumstances and how far the battery charges when the notebook is connected to an AC adapter. For example, you can use the following settings to ensure that the battery is only charged when its state of charge is less than 45%. In addition, the battery is only charged to 55%. Once this state is reached, the power adapter supplies the notebook with power, but does not charge the battery any further:

```
# in /etc/default/tlp oder /etc/tlp.conf
...
START_CHARGE_THRESH_BAT0=45
STOP_CHARGE_THRESH_BAT0=55
```

This type of battery tuning is not without controversy:

- For one, it is unclear how much the measure will really achieve.
- Second, the reduced charging can mess up the battery calibration. The battery or its control software (BIOS/EFI) may believe that the battery is no longer fully capable. The solution is usually to completely discharge and charge the battery once. You can trigger this recalibration as follows:

  ```
  user$ sudo tlp recalibrate
  ```

- Naturally, it's inconvenient when you unexpectedly need your notebook on the road to find that the battery is half empty. Before a planned trip, you should fully charge the battery as follows:

  ```
  user$ sudo tlp fullcharge
  ```

Problems in Practice

On my Lenovo P1, which runs most of the time in the office, I set the charging thresholds of 45 and 55% mentioned in the previous listing. After about a month, the notebook then suddenly no longer wanted to charge! As long as an AC adapter was connected, its power was used to run the notebook, but the battery charge kept dropping.

Even `tlp fullcharge` did not provide any improvement. Finally, I found a tip on the internet on how to reactivate the loading behavior via a hidden

reset button: *https://forums.lenovo.com/t5/ThinkPad-P-and-W-Series-Mobile/Thinkpad-P1-plugged-in-not-charging/td-p/4370761*.

Very strange—but since then, everything seems to be working again.

14.10.4 Fan Control

Nothing is more annoying than when the fan is constantly howling in an inherently quiet notebook. Can anything be done about that? In my experience, not so much; just make sure you choose a model that is quiet when you buy it! After your purchase, you can reduce the power consumption and thus the heat development a little by undervolting, depending on the CPU. Finally, you can rein in the CPU by disabling turbo mode as well as limiting the maximum clock frequency. Of course, this also limits the performance of the CPU. So it's a trade-off between speed and noise.

thinkfan

Basically, controlling the fan is only possible at all in relatively small number of notebook computers. The `thinkfan` program from the package of the same name presented here works with some Lenovo models. Before `thinkfan` can become active, the `thinkpad_acpi` kernel module must be loaded with the `fan_control=1` option. To do this, you need to create a new file in `/etc/modprobe.d` and then restart the computer:

```
# file /etc/modprobe.d/thinkpad_acpi.conf
options thinkpad_acpi fan_control=1
```

The configuration of `thinkfan` is done by the `/etc/thinkfan.conf` file. Basically, there are two things to set there: how `thinkfan` measures the temperature in the notebook and at what temperature the fan should be operated and at what strength. Tips for configuring the fan levels can be found here:

- *https://www.thinkwiki.org/wiki/Thermal_Sensors*
- *https://github.com/vmatare/thinkfan*

To experiment, start `thinkfan` with the `-n` option. This allows you to watch the program at work:

```
root# thinkfan -n
...
sleeptime=5, tmax=53, last_tmax=53, biased_tmax=53 -> fan="level 3"
sleeptime=5, tmax=52, last_tmax=53, biased_tmax=52 -> fan="level 2"
sleeptime=5, tmax=50, last_tmax=52, biased_tmax=50 -> fan="level 1"
...

Ctrl+C
```

When you are happy with your configuration, you can start `thinkfan` permanently:

```
root# systemctl enable --now thinkfan
```

In my tests, I didn't manage to regulate the fan better (i.e., make it quieter) with `thinkfan` than it was by default through the BIOS/EFI. This is possibly also because my notebook is equipped with *two* fans (one each for CPU and GPU), so the control model of `thinkfan` is inadequate.

Caution

By its very nature, manual fan control is done at your own risk. If the CPU or other components of the computer are regularly too hot, their service life will be reduced! For this reason, you should use fan control programs with caution and research the experiences of other users on the internet in advance.

14.11 Printing System (CUPS)

On Linux and macOS, the *Common UNIX Printing System* (CUPS, *https://cups.org*) is responsible for processing and buffering print jobs and for converting the print data to the printer's format. Because I have already briefly described the configuration of a printer in GNOME and KDE in Chapter 4, Section 4.4.4 and in Chapter 5, Section 5.4, I will go into more detail about the internal details of CUPS at this point. The section is of particular interest if you encounter problems configuring your printer or if you have special requests that cannot be met with the GNOME or KDE configuration tools.

Apple acquired the rights to CUPS in 2007 and took care of its maintenance until early 2020. Michael Sweet, the lead developer of CUPS, left Apple in 2020. The CUPS open-source code then found a new home in the OpenPrinting project (see *https://openprinting.github.io*).

CUPS as snap package

Currently, CUPS is contained in common packages that are installed by default on desktop distributions. This is supposed to change in Ubuntu 24.10. According to current plans, CUPS will be delivered in a snap package.

Printer purchase

As usual in the Linux environment, it's worth doing a little research before buying a printer. My advice is to use a laser printer: most models are well-suited for operation on Linux. However, make sure that the PostScript, PDF, or PCL formats are supported. Occasionally, there are still particularly cheap models that are not compatible with common standards and only work with Windows drivers.

For inkjet printers, the extent of Linux support is very dependent on the model. It works relatively well with many HP models. However, new printers are sometimes missing from the CUPS printer database. As a rule, inkjet printers from other manufacturers and very new models in general cause more problems.

Commercial drivers

Some printers for which there re no open-source drivers are supported by the commercial printer driver provided by TurboPrint (*https://www.turboprint.info/*). Also, with TurboPrint, you can get better results with some photo printers than with the standard CUPS drivers. But before you buy, take a look at the list of supported models and the version changes page.

14.11.1 Sequence of the Printing Process

Printer management on Linux is done by a network service. Any printer set up on Linux can be used by all other computers on the local network if configured accordingly.

PostScript and PDF

The entire printing philosophy on Unix/Linux was originally based on PostScript printers. *PostScript* is a programming language for describing page content. PostScript printers expect print data in this format. Almost all Linux programs with printing functions send PostScript data to the printing system.

For several years now, there has been a shift from PostScript to PDF format. Loosely speaking, a PDF is a compressed representation of a PostScript file with some additional features. Because more and more programs can now generate PDF files and more and more printers can process PDF files directly, CUPS is increasingly trying to avoid the intermediate PostScript step.

Spooling systems

In the early days of Unix/Linux, `root` could print by copying a PostScript file directly into a printer's device file. Now, apart from `root`, ordinary users also want to print—possibly even those who work on another computer on the network. And no one wants to be bothered with device names in the process. For this reason, so-called spooling systems exist. These systems have several tasks:

- They provide easy-to-use commands for printing, thanks to which it is not necessary to specify a device name when printing, but simply the printer name.
- Depending on the configuration, they allow all users to print, even on a network if required.
- They make it possible to connect multiple printers to one computer and manage them.

- If multiple print jobs arrive at the same time, the jobs are temporarily stored in print queues until the printer is available.
- Furthermore, spooling systems can perform various additional functions—for example, logging who prints how much, among others.

The most popular spooling system for Linux is CUPS. In the past, BSD-LPD or LPRng were used instead of CUPS, for example. Regardless of the spooling system, the command to print a file still looks the same:

```
user$ lpr -Pname file
```

Here, `name` is the name of the printer (strictly speaking: the name of the print queue). If you omit the `-P` option, the printout will be made on the default printer.

Printer filters (Ghostscript)

So far, I have assumed that you use a printer that supports PostScript or possibly even PDF directly. In practice, however, printers that are not PostScript- or PDF-enabled are often used. For such printers to work on Linux, it's necessary to convert the PostScript or PDF data to the respective printer format. Internally, the Ghostscript program is used for this purpose (`gs` command).

A so-called filter takes care of calling `gs`. This is a program (a script) that processes input data and provides output data. In particular, the filter for the printing process must pass the correct parameters to `gs`: the name of the printer model, the desired resolution, the desired page size, and so on. It converts the PostScript data page by page into bitmaps and passes it on—together with the print commands of the respective printer. Ghostscript also makes use of external printer drivers in its work.

The main driver project for Linux is Gutenprint (formerly GIMP-Print).

Document filter

Now, PostScript and PDF are the preferred formats of all CUPS print files, but sometimes you just want to print a text or graphics file. Of course, you can load the text file into an editor, which will then pass the file to the printing system in PostScript or PDF format. Likewise, you can convert the graphics file to PostScript or PDF formats using a graphics program or converter.

However, it's even more convenient to simply run `lpr file` for such files as well. For this to work, the spooling system tries to detect the type of file to be printed. If that's successful, the file is converted to PostScript or PDF format using suitable programs.

All together

Say that you have correctly configured an inkjet printer on your computer. The printer name is `pluto`. Now you want to print the `mypicture.png` graphics file and execute the following command:

```
user$ lpr -Ppluto mypicture.png
```

The following operations now take place:

- `lpr` passes the file to the CUPS spooling system.
- The spooling system passes the file to the filter system.
- The filter detects the file type (PNG) and converts the bitmap to a PostScript or PDF file, depending on the CUPS version.
- This file is passed to Ghostscript, which converts it to the manufacturer-specific format of the printer, `pluto`.
- Once the printer (`pluto`) has processed all previously started print jobs, it prints `mypicture.png`.

14.11.2 Internal Details of CUPS

Configuration files

The `cupsd` background service is started by systemd. On older printing systems, almost the entire printer configuration was done through the `/etc/printcap` file. With CUPS, on the other hand, this file plays almost no role anymore. It's still available for compatibility reasons, but it only contains a list of all known queues (without any additional parameters). The actual CUPS configuration is done by the files of the `/etc/cups` directory. Table 14.6 provides an overview of the most important files.

File	Contents
`classes.conf`	Definition of all classes
`cupsd.conf`	Central CUPS configuration file
`lpoptions`	Changes compared to the basic configuration
`printers.conf`	Definition of all printers
`ppd/name.ppd`	Configuration for the `name` queue
`.cups/lpoptions`	Personal settings (KDE)

Table 14.6 Configuration Files in /etc/cups

In `cupsd.conf`, various installation directories are set, the port of the CUPS daemon for the *Internet Printing Protocol* (IPP), the options for printer browsing, security parameters, access rights for clients on the network (*allow/deny*), and so on. The `/etc/cups/ppd` directory contains the corresponding PPD file for each printer name contained in `printers.conf`. The printer model and driver, paper size, and required resolution are stored in it.

If the system administrator (`root`) changes printer options or settings, like the sheet size, print resolution, portrait or landscape format, or others, then these changes will be saved in the `lpoptions` file. The changes apply to all

users who have not already made changes themselves. These user-specific changes are stored in `.cups/lpoptions`.

> **"If It Ain't Broke, Don't Fix It"**
>
> CUPS is a complex system, and you should only use the tools provided for the purpose of configuration. Manual changes to the configuration are only recommended for CUPS professionals. More details about the CUPS configuration can be found at *https://www.cups.org/documentation*.

MIME The `/usr/share/cups/mime/mime.types` file contains a list of all document types that are automatically recognized by CUPS and converted to PostScript or PDF files. The `mime.convs` file stored in the same directory specifies which filter you should use. The specified filters must be located as executables in `/usr/lib/cups/filter`.

PPD files For CUPS, every printer looks like a PostScript or PDF printer. Printer-specific details such as the size of the nonprintable margin, the printer resolution, commands for certain additional functions such as the paper feed, special features such as duplex printing, and so on are stored in *PostScript Printer Definition* (PPD) files. The PPD format was defined by Adobe and is also used on Windows and macOS.

As not every printer is actually a PostScript or PDF printer, CUPS PPD files also contain the required Ghostscript commands as comments, including all options, so that `gs` can convert the print data into the printer format. The following lines show some excerpts from a PPD file for an inkjet printer:

```
*PPD-Adobe: "4.3"
...
*Manufacturer: "HP"
*ModelName: "HP Deskjet 6980 Series hpijs"
*FoomaticIDs: "HP-DeskJet_6980 hpijs"
*FoomaticRIPCommandLine: "gs -q -dBATCH -dPARANOIDSAFER -dQUIET
-dNOPAUSE -sDEVICE=ijs -sIjsServer=hpijs%A%B%C -dIjsUseOutputFD%Z
-sOutputFile=- -"
...
```

This information comes from the `ppds.dat` database, which contains PPD entries for all printers known to CUPS. Depending on the distribution, the binary `ppds.dat` file can be located, for example, in the `/var/cache/cups` directory. If your printer isn't included in this database and you can't find a compatible model either, a suitable `*.ppd` file from the internet may help.

During the printing process, CUPS extracts the Ghostscript parameters for the required printer from the `*.ppd` file, uses them to call `gs`, and thus converts the PostScript data into the format of the respective printer. The resulting data is then sent to the printer device.

HPLIP

HP itself develops free printer drivers for many of its printers, scanners, and multifunction devices as part of the *HP Linux Imaging and Printing* (HPLIP) project. The GPL is predominantly used as a license; sometimes the MIT or BSD license is used instead. HP is a shining example in the computer industry of active open-source support.

Because many HP printers are directly supported by CUPS even without HPLIP, the use of the HPLIP functions is mostly optional. You can find more information about HPLIP at *https://developers.hp.com/hp-linux-imaging-and-printing*.

For HPLIP, there is a graphical user interface, `hplip-toolbox`, which is in a separate package on many distributions (such as `hplip-gui` on Ubuntu) and can be installed separately. The program independently detects connected HP devices and helps with their configuration and use. `hp-toolbox` can, among other things, display the ink cartridge fill level of many HP inkjet printers—a function that CUPS does not offer by itself.

Classes

Classes help you to set up a printer pool on large networks. A printout directed to a class is then made on the first available printer of this pool.

IPP

CUPS supports the IPP (see *https://pwg.org/ipp*). This protocol greatly simplifies the use of printers on the network across Linux boundaries. IPP is supported by all common operating systems.

Spooling

All data sent to the printer is cached in the `/var/spool/cups/*` directory until the printout is complete. Note that spool data is not lost even if Linux is rebooted. After the restart, `cupsd` detects that there are files that have not yet been printed and will continue to try to transfer the data to the printer.

14.11.3 CUPS Web Interface

Basically, it's possible to modify the CUPS configuration files in a text editor. However, because of the great complexity, this is rarely recommended. Usually, it makes more sense to use local configuration tools, such as the system settings of KDE or GNOME or the SUSE-specific YaST program.

For server installations without a graphical user interface, you can use the CUPS web interface for configuration (see Figure 14.6). For security reasons, this interface is only available on the local computer. The following address takes you to the home page: *http://localhost:631*. It's somewhat more difficult to use the web interface on another computer. It's possible to modify

the /etc/cups/cupsd.conf configuration file accordingly, but in practice this often entails problems and security gaps.

Using the CUPS web interface via SSH

A much simpler way is provided by the ssh command, which you can execute in a terminal window on the local computer. By using the -L option, you can redirect port 631 of the machine running CUPS to any port on the local machine. As long as the SSH connection is established, you can use the CUPS web interface on the local machine in a web browser, using the port number specified in the SSH command. Thus, in the following example, port 631 of the CUPS computer is redirected to port 6310 of the local computer (localhost):

```
user@local-machine# ssh -L 6310:localhost:631 user@cups-hostname
```

In the web browser on your local computer, you can now enter the following address: *http://localhost:6310*. To use the admin parts of the web interface, you must log in with a user name for the CUPS computer and the corresponding password. Now you can set up new printers, manage print jobs, and so on.

When you're done, close the SSH connection using Ctrl+D.

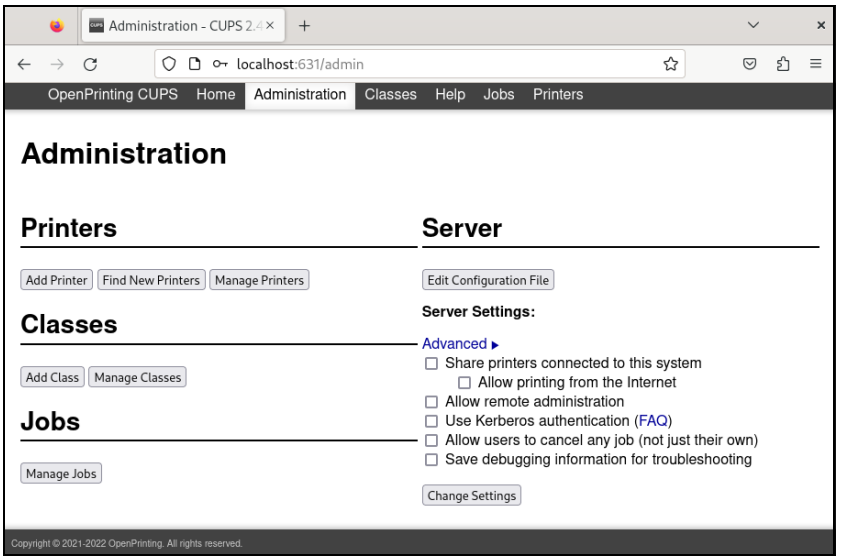


Figure 14.6 CUPS Configuration in Web Browser

14.11.4 Administrating CUPS Using Commands

Usually you use the dialogs of a given program for printing and the corresponding tools of GNOME or KDE for managing the print jobs. Command line users can also choose from various commands to print files or manage

print jobs. These commands are especially useful if you want to automate printing tasks using script files.

lpr

You can use `lpr` to print a file. If you have set up multiple printers, you want to use the `-P` option without spaces to specify the queue name. You can omit `-P` for the default printer:

```
user$ lpr -P<name> file
```

If a print file already has a printer-specific format, you can pass the additional `-l` option to `lpr`. The command then bypasses the usual filter system and sends the printer data unchanged to the printer. For example, if you want to print a PostScript file on a PostScript printer, this can save a lot of time.

Using a pipe, `lpr` can also be used to print the output of another command. The following command prints the file list determined with `ls` on the default printer:

```
user$ ls -l *.png | lpr
```

Instead of `lpr` you can also use the `lp` command (see `man 1 lp` for the syntax). This command is intended to make life easier for those switching from conventional Unix printing systems.

lpq, lprm

All print jobs that cannot be executed immediately are buffered, with a separate queue for each printer set up. You can view the contents of the queue using `lpq -P<name>`.

Print jobs that you have initiated yourself can be deleted again via `lprm -P<name> <id>`, specifying the name of the queue and the ID number of the job. You can determine the correct number in advance by using `lpq`:

```
user$ lpq
FS-1800+ is not ready
Rank    Owner   Job  File(s)                        Total size
1st     kofler  20   evince-print                   17408 bytes
2nd     kofler  21   evince-print                   16384 bytes
user$ lprm 20
user$ lprm 21
```

lpc

`lpc` allows for a finer control of the printing process. After starting this command, you are in an interactive working environment where you execute commands like `status`, `help`, and so on. `topq` changes the position of a print job in the queue. Enter the printer name and the job number as parameters. Some of the commands in `lpc` (like `topq`) may only be executed by `root`; `exit`, `bye`, or `quit` terminates `lpc`.

lpstat, lpinfo, lpadmin, lpoptions

`lpstat` provides information about all printers available for CUPS. `lpinfo` retrieves a list of all available print devices and printer drivers. You can use `lpadmin` to set up a new printer or delete an existing printer configuration. `lpoptions` displays or changes CUPS printer options:

```
user$ lpoptions -o PageSize=A4
```

14.12 Logging (Syslog)

The kernel, various administration tools (PAM, APT, dpkg) and most network services log events and errors to countless files in `/var/log`. These logging files are extremely useful during the startup of a new service to find configuration errors. When a server is running, logging files can provide clues to security issues.

Syslog

To avoid the need for each program to implement its own logging functions, the kernel and a number of admin tools and server services make use of central logging functions, usually referred to as *syslog*. There are various implementations of syslog; the most popular one currently is `rsyslogd`.

However, not all network services use syslog. In particular, the "big" server services, such as Apache, CUPS, MySQL, and Samba, each use their own logging functions built into the program. They thus escape the global syslog configuration. The logging parameters are rather set in the respective configuration files of the program.

Journal

Fedora and current RHEL distributions only use the journal function of systemd for logging tasks instead of `rsyslogd`. `rsyslogd` can also be installed if the permanent storage of logging files as text is preferred. In most other distributions, `rsyslogd` runs by default, but even in this case the journal is active in parallel.

Unlike `rsyslogd`, the journal uses a binary file format that is protected against subsequent changes. More details about the journal follow in Section 14.13.

14.12.1 rsyslogd

Configuration

In many distributions (Debian, SUSE, and Ubuntu), the syslog services are implemented by default by the `rsyslogd` program. Its configuration is carried out by the `/etc/rsyslogd.conf` and `/etc/rsyslog.d/*.conf` files. Ahead, I describe the configuration on Ubuntu as an example. There, most of the settings are located in `/etc/rsyslog.d/50-default.conf`.

The syslog configuration files contain rules that consist of two parts:

- **Selector**
  The first part of each rule specifies what is supposed to be logged.
- **Action**
  The second part controls what should be done with the message.

Rules can be spread across multiple lines using the \ character. It's possible that multiple rules apply to one message. In this case, the message is logged or passed on several times.

Selector

Each selector consists of two parts separated by a dot: *service.prioritylevel*. You can specify multiple selectors separated by a semicolon. Furthermore, multiple services can be separated by commas in *one* selector. All Linux programs that use syslog must assign a service and priority to their messages.

Syslog distinguishes between the services (*selectors*) grouped in Table 14.7 when logging. `auth` and `authpriv` apply to user-level and system-level authentication system messages, respectively, such as the notification of an incorrect login, for example. In reality, only `authpriv` is relevant most of the time. However, the messages logged in this way are very sensitive from a security point of view. The resulting logging files are usually set up in such a way that they can only be read by `root`.

Selector	Meaning
`auth`	Authentication
`authpriv`	Authentication (privileged)
`daemon`	Various background services
`ftp`	FTP
`kernel`	Kernel messages
`lpr`	Printing system (CUPS)
`mail`	Mail server
`news`	Usenet server
`syslog`	Messages about the syslog service itself
`user`	Default, if no selector is specified
`uucp`	UUCP (Unix to Unix copy; deprecated)

Table 14.7 Syslog Selectors

Selector	Meaning
local0 through local7	For free use
*	Applies to all services

Table 14.7 Syslog Selectors (Cont.)

Syslog also knows the following priority levels (in increasing importance): debug, info, notice, warning = warn, err = error, crit, alert, and emerg = panic. The warn, error, and panic keywords are considered obsolete; you should use warning, err, and emerg instead. The * character includes all priority levels. The none keyword applies to messages that are not assigned a priority.

Specifying a priority level includes all higher (more important) priority levels. Thus, the mail.err selector also includes crit, alert, and emerg messages from the mail system. If you explicitly want only messages of a certain priority, you must prefix the = character (i.e., mail.=err).

Action The action usually has the name of a logging file. By default, logging files are synchronized after each output. If the filename is preceded by a minus sign, syslog will not synchronize. This is much more efficient, but messages that haven't yet been physically saved will be lost in the event of a crash.

Syslog can also forward messages to FIFO files or pipes. In this case, you need to prefix the file name with the | character. The * character means that the message will be sent to all users logged into consoles or via SSH. Because this is very annoying, this logging method is only used for critical messages. For more details on the syntax of rsyslog.conf, see the man page of the same name.

Example The following lines reproduce the syslog configuration of Ubuntu, slightly shortened and formatted a bit more clearly:

```
# file /etc/rsyslog.d/50-default.conf bei Ubuntu (shortened)
# Selector                      Action
auth,authpriv.*                 /var/log/auth.log
*.*;auth,authpriv.none          -/var/log/syslog
kernel.*                        -/var/log/kernel.log
mail.*                          -/var/log/mail.log
mail.err                        /var/log/mail.err
*.emerg                         :omusrmsg:*
```

In plain language, the above configuration means the following:

- /var/log/auth contains authentication messages of all priority levels. These include failed and successful login attempts (also via SSH), PAM

messages, sudo commands, and so on. As the only logging file, auth is synchronized immediately with each message.

- /var/log/syslog contains *all* messages logged via syslog (including authentication messages that are not assigned a priority). The all-encompassing approach is both an advantage and a disadvantage. On the one hand, this allows you to extract all conceivable information from a single file. On the other hand, it is of course particularly difficult to find relevant entries in this hodgepodge.
- /var/log/kernel.log contains all kernel messages.
- The messages of the mail system (such as Postfix, Dovecot, SpamAssassin) are distributed via two files. In mail.log, *all* messages are stored; in mail.err, only error messages.
- Critical system messages, such as about an imminent shutdown or about kernel errors, are forwarded by :omusrmsg:* to all users—more precisely, to all terminal windows and consoles. omusrmsg is a rsyslog module to send messages to users.

For changes to the syslog configuration to take effect, the syslog service must be restarted:

```
root# systemctl restart rsyslog
```

logger

You can use the logger command to record syslog messages yourself in a script or to test new syslog rules. Typically you use the command as follows:

```
user$ logger -t mydaemon -p authpriv.info "xxx has a new password"
```

Instead of mydaemon, you can specify any keyword that will help you search the logging files later. The -p option is used to specify the selector. Syslog automatically adds time information to your message. In the relevant logging file, the entry created earlier then looks as follows:

```
Jun 16 07:55:03 localhost mydaemon: xxx has a new password
```

14.12.2 Kernel Logging

Messages from the kernel are written to a 16 KiB ring buffer in RAM. When this buffer is full, old messages are deleted to make room for new messages. You can view the contents of this ring buffer by using dmesg. If you specify the -c option in the process, the ring buffer will be emptied at the same time.

All kernel messages are also written to the virtual /proc/kmsg file. This file is used to pass the kernel messages to syslog.

The dmesg kernel messages are often prefixed with a time in the format [nnn.nnnnnn]. The number before the dot indicates the number of seconds since the system start; the other six digits specify the time to millionths of a second. When storing the kernel messages in a logging file, this time information is usually supplemented by the absolute time.

With dmesg -T, dmesg displays absolute times. It should be noted that these times may be incorrect if the computer has been idle in the meantime or if the execution of a virtual machine has been paused.

14.12.3 System Startup Log

In some distributions, the screen outputs of the init system (see Chapter 20) are logged in the /var/log/boot.log file. Almost all distributions use systemd as the init system. All systemd output is then also logged in the journal (Section 14.13). These outputs are much more detailed, but often also more confusing than the conventional boot.log file.

14.12.4 Logrotate

Logging files are gradually getting bigger and bigger. To keep the memory requirements of the logging files under control, using logrotate is a good choice. This program is called once a day by the Cron script /etc/cron.daily/logrotate in many distributions. It then processes all logging files described in the configuration files in /etc/logrotate.d. The way logrotate handles the logging files depends in detail on the respective program configuration. However, the basic procedure is always the same and looks as follows:

- Logrotate renames the current logging file so that name becomes name.0.
- Logrotate creates a new, empty logging file called name.
- For many server services, Logrotate prompts the daemon to reload the configuration by service name reload. On this occasion, the daemon recognizes that there is a new, empty logging file and now uses it.
- Logrotate compresses name.0 or name.1 (delaycompress option). delaycompress avoids conflicts between the daemon, which may still write to name.0, and the compress command.
- Logrotate renames already existing logging archives so that name.4.gz becomes name.5.gz, name.3.gz becomes name.4.gz, and so on. This process is called *rotating* and gives the package its name.
- If there are more than a given maximum number of logging archives, the oldest archive files will be deleted.

Configuration

The /etc/logrotate.conf file contains some default settings for Logrotate. These settings apply only if the program-specific configuration files do not contain different data. /etc/logrotate.d contains detailed settings for various programs that produce logging files. These files do not come from the Logrotate package, but from the packages of the respective program. Thus, the samba package provides, for example, /etc/logrotate.d/samba. This ensures that the files match the respective installed program version and that Logrotate informs or restarts the respective server service about the renaming of the logging files.

The following lines show the logrotate configuration for Apache on Ubuntu as an example. Here, logrotate processes the logging files daily. The logging files are renamed and compressed. This also resets the access rights so that the logging files can only be read by root as well as by members of the adm group. The archive is limited to 14 files; that is, you can only access the logging data of the preceding two weeks if required. Provided Apache is running, it's informed by reload that there are new logging files (invoke-rc.d is actually an old System V init script, but it contains code to make it compatible with systemd):

```
# file /etc/logrotate.d/apache2 (Ubuntu 21.04)
/var/log/apache2/*.log {
  daily
  missingok
  rotate 14
  compress
  delaycompress
  notifempty
  create 640 root adm
  sharedscripts
  prerotate
    if [ -d /etc/logrotate.d/httpd-prerotate ]; then
      run-parts /etc/logrotate.d/httpd-prerotate
    fi
  endscript
  postrotate
    if pgrep -f ^/usr/sbin/apache2 > /dev/null; then
      invoke-rc.d apache2 reload 2>&1 | logger -t apache2.logrotate
    fi
}
```

14.12.5 Logwatch

Automatic log analysis

The manual reading of logging files may be quite interesting at first or during a search for a bug. However, after a short time, like any other server

administrator, you will lose the desire to study logs and will neglect your logging files more and more. Automated tools for logging analysis are intended to provide a solution in this case. As an example, I present the `logwatch` program, which is included as a package in most distributions.

After installation, `logwatch` is executed once a day by `/etc/cron.daily/*logwatch`. It analyzes the entries of the logging files for the preceding 24 hours, summarizes the relevant information in an email, and sends it to `root`. (`logwatch` therefore requires that an email server is running on your computer in order to send the logging summary!) The logging summary is usually several pages long—actually too long to read completely every day. The amount of text depends on how many server services are installed and how many events have been logged during the preceding day. The following lines show a `logwatch` email that has been heavily abbreviated for space reasons:

```
---------------------- Dovecot Begin ------------------------
Dovecot IMAP and POP3 Successful Logins: 159
Dovecot disconnects: 146

--------------------- fail2ban-messages Begin -----------------------
-
 Banned services with Fail2Ban:    Bans:Unbans
    sshd:                          [953:948]

---------------------- httpd Begin ------------------------
 A total of 57 sites probed the server
    108.62.x.y
    176.95.x.y
    178.189.x.y
    ...

--------------------- pam_unix Begin ------------------------
dovecot:
   Authentication Failures:
       postmaster: 65 Time(s)
       info: 59 Time(s)
       test: 59 Time(s)
       admin: 58 Time(s)
       ...

sshd:
   Authentication Failures:
       root (58.242.x.y): 440 Time(s)
```

```
      unknown (82.185.x.y): 79 Time(s)
      ...

 -------------------- Postfix Begin -----------------------
    2188   SASL authentication failed                  2,188
     220   Miscellaneous warnings                        220
   2.944M  Bytes accepted                            3,086,942
   2.081M  Bytes sent via SMTP                       2,182,092
1021.312K  Bytes sent via LMTP                       1,045,824
       6   4xx Reject relay denied                      17.65%
      28   4xx Reject unknown user                      82.35%
     ...

     --------------------- Disk Space Begin -----------------------

/dev/md1        488M  166M  296M  36% /boot
/dev/md2        2.0T  112G  1.8T   6% /
/dev/md3        1.7T  107G  1.5T   7% /local-backup
```

Configuration

You may be wondering why Logwatch works as if by magic without any configuration. The reason is simple: Together with Logwatch, a default configuration is installed in the `/usr/share/logwatch/default.conf` directory. The `logwatch.conf` file located there contains some global basic settings. There you can set, for example, to whom the summary should be sent (`MailTo = root`), for which period the logging files should be analyzed (`Range = yesterday`), how detailed the summary should be (`Detail = Low`), and so on.

In addition, the `default.conf/services/*.conf` files contain configuration settings for numerous server services, such as Apache, Postfix, ClamAV, Dovecot, Sendmail, SSH, and so on. Of course, these configuration files only take effect if the services in question are actually running. Moreover, you will get relevant results only if you haven't changed the locations of the default logging files.

The `/usr/share/logwatch/dist.conf` directory is for distribution-specific changes from the default configuration. Settings made here take precedence over the default configuration.

To change the configuration yourself, you need to copy the relevant file to the appropriate directory in `/etc/logwatch` and modify it there. It's sufficient for this file to contain only the changes from the original.

To try this out, you can then run Logwatch manually:

```
root# logwatch --mailto name@host
```

14.13 Logging (Journal)

The systemd init system, described in Chapter 20, Section 20.1, contains its own logging functions, called the *journal*. The `systemd-journald` background process is responsible for logging. Compared to traditional syslog services, there are three fundamental differences:

- The journal is stored in a binary format to save space. However, this format is also the biggest disadvantage of the journal: countless tools assume that the logging files are available in text format and can be easily analyzed using `grep`.
- The journal is protected against subsequent changes. This makes it impossible for an attacker to remove his or her traces.
- The entire journal is stored in one place. The separation into multiple logging files, which is usual with syslog, is omitted (and with it also the corresponding configuration). To filter specific messages from the journal, you must pass appropriate options to `journalctl`. (Details will follow ahead.)

Apart from that, however, the journal is syslog-compatible. All services that previously used syslog for logging can use the journal without any changes. The `logger` command described in the previous section also cooperates with the journal without any problems.

Presumably, the journal will completely replace `rsyslogd` in more and more distributions in the future. Currently, this is already the case with Red Hat–based systems (i.e., Fedora, RHEL from version 9 on, and their clones) and Debian (from version 12). In most other distributions, `rsyslogd` and the journal run in parallel. Depending on the configuration, certain data is then simply logged twice. This is what happens on Ubuntu, for example.

There is a simple reason that the switch to the technically superior journal is progressing so slowly: various tools such as Fail2Ban or Logwatch rely on the fact that the logging files are saved as text files in `/var/log` and can be easily analyzed. This means that in distributions with the journal, `rsyslogd` often needs to be added.

System versus user

Both systemd and the journal act in a dual manner at the system and user levels. Especially desktop systems like GNOME make intensive use of systemd and the journal. The user instance of the journal then logs messages from the display manager `gdm`, `dbus-daemon`, and local programs (e.g., backup services).

Logging files

If the `/var/log/journal` directory exists, the journal stores its logs there, both system messages and user-specific messages. If this directory doesn't exist, `/run/log/journal` will be used.

At first glance, this does not seem to be a big difference—but /run is usually a temporary file system. Logs stored there are therefore lost with every restart! Persistent storage therefore requires /var/log/journal to exist. There is currently no consensus on this, with persistent storage on Arch Linux, Debian, Fedora, and Ubuntu, and no persistent storage on RHEL and its clones, SUSE, and openSUSE.

Storing the journal permanently

If you want to store the journal permanently on RHEL or SUSE distributions, you can simply create the /var/log/journal directory and restart the journal services:

```
root# mkdir /var/log/journal
root# chown root:systemd-journal /var/log/journal
root# systemctl restart systemd-journald
root# systemctl restart systemd-journald.socket
root# systemctl restart systemd-journal-flush
```

14.13.1 journalctl

The binary log files can be read using the journalctl command. If you start the command without options, it displays all available messages at the system and account levels, with the oldest message first. Then you can scroll through the log as you would using less.

As with less, pressing [>] jumps to the end of the log—that is, to the latest messages. Note, however, that this process can take a very long time if persistent message storage is enabled and there are possibly millions of messages available. In such cases, you should make sure to use the following options to filter the flood of messages in advance:

- -b shows only the messages since the last reboot of the computer.
- -f starts journalctl in continuous operation, constantly displaying the currently arriving messages. [Ctrl]+[C] terminates the command.
- --facility name shows only messages for a specific syslog group (such as --facility authpriv).
- -k shows only kernel messages.
- --list-boot returns a list of all machine boots (requires persistent journal storage).
- -n N shows only the last N lines.
- -p priority displays only messages up to the specified priority level. The following numerical values or character strings are allowed instead of priority: 0 = emerg, 1 = alert, 2 = crit, 3 = err, 4 = warning, 5 = notice, 6 = info, and 7 = debug.

- `-t search-keyword` shows only messages for the specified syslog keyword (tag, as for `logger -t`).
- `-u name` shows only messages for the specified systemd service (unit name, such as `avahi-daemon`).
- `--user` displays only user-specific messages (relevant only on desktop systems).
- `--system` displays only system messages.

Examples The following command filters out all logging messages since the last reboot that contain the search term `systemd`:

```
root# journalctl -b | grep systemd
...
... systemd[1]: Starting Hostname Service...
... dbus-daemon[2240]: [session uid=1000 pid=2240] \
      Activating via systemd: ...
... systemd[2214]: Starting Tracker metadata extractor...
... sshd[647884]: pam_systemd_home(sshd:auth): \
      systemd-homed not available ...
... systemd-logind[726]: New session 7 of user kofler.
```

The next command displays the last 30 messages at the account level:

```
user$ journalctl --user -n 30
... gnome-shell[2364]: Window manager warning: \
      Ping serial ... was reused ...
... gsd-media-keys[2619]: gvc_mixer_card_get_index: \
      assertion xxx failed
... appindicatorsupport@xxx.gmail.com[2364]: \
      Using Brute-force mode ...
... okular[214197]: Unsupported return type 65 \
      QPixmap in method "grab"
...
```

14.13.2 Configuration

The journal is controlled by `/etc/systemd/journald.conf` and the following configuration files:

/etc/systemd/journald.conf.d/.conf*
/run/systemd/journald.conf.d/.conf*
/usr/lib/systemd/journald.conf.d/.conf*

Details about the settings allowed there can be found with `man journald.conf`. Here, I want to focus on a few selected parameters:

- `ForwardToSyslog` specifies whether messages logged by the journal should also be passed to a traditional syslog service.
- `MaxFileSec` specifies after what time at the latest a new logging file should be started. The default setting is one month. This setting is only relevant if the logging files grow slower than the `SystemMaxUse`, `SystemMaxFileSize`, and `SystemKeepFree` limits specify.
- `MaxLevelStore` specifies the priority level up to which messages are stored in the journal. The default setting is `debug`.
- `SystemMaxUse` specifies the maximum percentage of the file system that the logging files can occupy. Before this limit is exceeded, old logging files will be deleted. The default setting is 10%.
- `SystemMaxFileSize` specifies the maximum size of a logging file. The default setting is one-eighth of `SystemMaxUse`. This results in automatic rotating, creating a maximum of seven older files in addition to the current logging file.
- `SystemKeepFree` specifies the percentage of the file system that must remain free. The default setting is 15%.

Reducing the journal space

If you want to reduce the space required in the `/var/log/journal` directory, you can use the following command to delete old logging data:

```
root# journalctl --vacuum-size=500M
```

14.14 Cockpit

Cockpit is a web interface for server monitoring and administration tasks (see Figure 14.7). In recent years, the open-source project has been driven forward with the significant involvement of Red Hat employees. For more information, see the following web pages:

- *https://cockpit-project.org*
- *https://github.com/cockpit-project/cockpit*

Although the program is operated via a web browser, Cockpit does not require the installation of a dedicated web server such as Apache or NGINX. Instead, all the necessary web server functions are integrated directly into Cockpit.

Apart from that, Cockpit tries to make the best possible use of the existing infrastructure. Cockpit is therefore a comparatively lean project that also consumes few resources during operation.

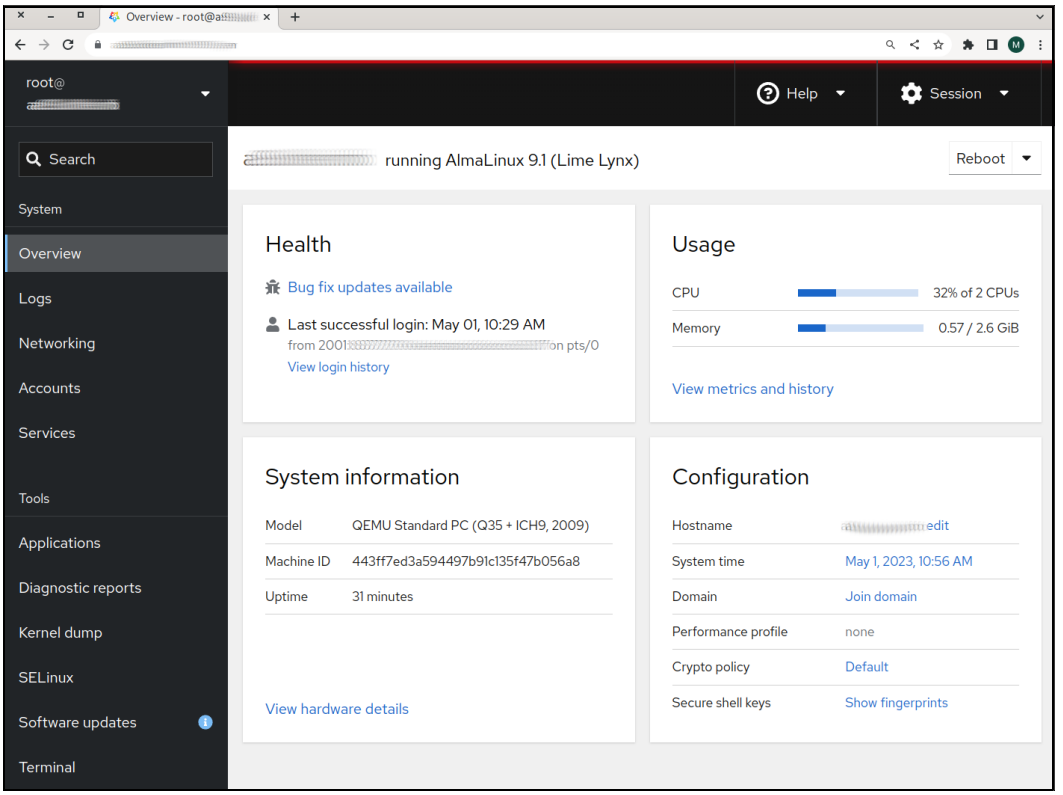


Figure 14.7 System Administration Using Cockpit

14.14.1 Installation

Cockpit in RHEL

Cockpit received its accolades with RHEL. In a standard installation of this distribution from version 8, Cockpit is preinstalled but not yet activated for the time being. However, this can be accomplished in no time at all:

```
root# dnf install cockpit          (only required for minimal installations)
root# systemctl enable --now cockpit.socket
```

Pay attention to the `.socket` extension! The `systemctl enable --now cockpit` command without the suffix doesn't work because cockpit is not started by systemd as an ordinary service, but as a socket.

Other distributions

The program is also available as a package for many other Linux distributions and can be installed using `apt/dnf/zypper install cockpit`. If necessary, you must enable port 9090 (in RHEL, this port is free by default for the default firewall zone):

```
user$ sudo firewall-cmd --add-service=cockpit
user$ sudo firewall-cmd --add-service=cockpit --permanent
```

You can find more tips on commissioning Cockpit at *https://cockpit-project.org/running.html*.

14.14.2 Security Concerns

Cockpit is without a doubt a practical tool with many functions. However, before you install or activate Cockpit, you should be aware of some possible security implications.

Cockpit provides a simple login on its web interface by default, with the same login data as for Linux accounts. For all distributions except Ubuntu, a `root` login is also allowed. Even if the login is not as `root`, users with admin rights have unrestricted rights even in Cockpit thanks to `sudo` and PolicyKit (`pkexec`).

Target for password crackers

As it becomes more widespread, Cockpit will establish itself as an attractive target for attack. Thus, automated password-cracking tools will try to guess a working combination of login name and password. This danger also affects SSH. With SSH, however, it is common to use Fail2Ban (see Chapter 23, Section 23.3) or comparable tools to block an attacker's IP address for a while after several unsuccessful attempts. Such an approach is also possible for Cockpit, but it requires manual work. Fail2Ban does not currently contain any preconfigured rules for Cockpit.

Cockpit itself provides only minimal protection against password crackers by default: a maximum of 10 simultaneous unauthenticated connections is allowed. This limits the speed of password crackers but does not prevent further login attempts. (The `MaxStartups` parameter in `/etc/cockpit/cockpit.conf` decides how Cockpit should react in case of many failed connection attempts.)

Behind the firewall

Provided it's using a server on the local network, one possible security measure is to block port 9090 to the outside. Naturally, this contradicts the idea of remote server maintenance.

14.14.3 Configuration

cockpit.conf

Some basic settings of Cockpit can be changed in the `/etc/cockpit/cockpit.conf` file. However, the concept of Cockpit is that the program should ideally run without any configuration. Consequently, `cockpit.conf` does not exist in many distributions. So if you want to make changes, you need to set up the file again. The few setting options are documented in `man cockpit.conf`.

HTTPS certificate By default, Cockpit uses a self-signed certificate for SSL encryption. The web browser then displays a warning that the connection is insecure. (Strictly speaking, the encryption is quite secure. However, the web browser cannot trust the certificate because it isn't signed by any known certificate authority.) The solution here is to use "real" certificates, such as those from Let's Encrypt. I describe how to set this up in detail in Chapter 24, Section 24.3. Now you need to instruct Cockpit to use these certificates. To do this, you should set up symbolic links from `/etc/cockpit/ws-certs.d` to your certificate files and replace `/etc/acme/my-hostname` with the installation location of your certificates:

```
root# cd /etc/cockpit/ws-certs.d
root# ln -s /etc/acme/my-hostname.cert my.cert
root# ln -s /etc/acme/my-hostname.key  my.key
root# systemctl restart cockpit.socket
```

Port Cockpit port 9090 is predefined in the systemd configuration file, `/usr/lib/systemd/system/cockpit.socket`.

14.14.4 Operation

Login Unless you log in as `root`, the web page displays a *Restricted Access* notice. You can therefore only use functions in Cockpit for which you do not need to have `root` privileges. If you have `sudo` rights, clicking the message text will help; this requires you to enter your password again. Cockpit remembers this password and passes it to `sudo` or `pkexec` if needed.

Basic functions After login, various dialog sheets provide information about the current state of the system. You can also read logging files or the journal, view and modify the configuration of the network and file systems, set up users, perform updates, start and stop systemd services, and so on.

Additional functions Some Cockpit functions are only available if the corresponding extension modules are explicitly installed. A list of all packages is provided by the following command:

```
root# apt/dnf list 'cockpit-*'
```

The following list points out some packages that are particularly important:

- `cockpit-composer`
  Creating disk images using `lorax-composer`
- `cockpit-file-sharing`
  Samba configuration

- `cockpit-podman`
  Managing containers using `podman`
- `cockpit-machines`
  Managing virtual machines (`virtlib`, see Chapter 32)
- `cockpit-storaged`
  Managing file systems

Chapter 15
Network Configuration

This chapter describes how you can connect your Linux computer to a local network or wireless network. For desktop installations, this can usually be done effortlessly using NetworkManager. The network configuration is somewhat more difficult for server installations, where often no DHCP server is used and therefore a manual configuration must be performed. This chapter explains the basics of network configuration (including IPv6) and how to perform a static network configuration without graphical configuration tools.

15.1 NetworkManager

NetworkManager is the most popular tool for LAN, Wi-Fi, and VPN configuration on desktop systems. (An exception is Raspberry Pi OS. This distribution uses its own configuration tools.) On RHEL, NetworkManager also handles the network configuration for server systems.

A background process is responsible for the basic functions of NetworkManager, which is started when the computer is booted. The NetworkManager user interface looks different depending on the desktop. I will focus on the GNOME variant in this section. On KDE, the dialogs look a little different, but the operation is quite similar.

Prerequisites for this chapter

NetworkManager works only if the program has control over the network interfaces. Most common desktop distributions are configured for this purpose by default:

- On Debian, `/etc/network/interfaces` may only contain settings for the loopback interface, `lo`. Interfaces that are to be controlled by NetworkManager (typically `ethN`, `enpXXX`, `wlanN`, or `wlpXXX`) must not be configured by this file!
- Fedora, RHEL, and the like fully rely on NetworkManager in current versions. Other configuration paths are no longer provided.
- On SUSE, the YaST module at **System • Network Settings** controls how the network configuration is done. To use NetworkManager, the **NetworkManager Service** option must be enabled in the **Global Options** dialog. This is the case by default for notebook installations. The alternatives are **Wicked Service** (a SUSE-specific network control service specifically for servers) or **Network Services Disabled** (manual configuration).

- For Ubuntu desktop installations, `/etc/netplan/01-network-manager-all.yaml` specifies that NetworkManager is given control over all network interfaces.

Disabling NetworkManager

If you configure your computer as a server or router, some distributions recommend that you disable NetworkManager and perform a purely static network configuration. For tips on how to do this, see the beginning of Section 15.4.

15.1.1 Configuration

In most common distributions, an icon in the menu bar or panel shows the current network state. Clicking this icon opens a menu that lists the active connection and all accessible or preconfigured networks (see Figure 15.1). If necessary, you can change the configuration via this menu or in the **Network** module of the system settings (see Figure 15.2).

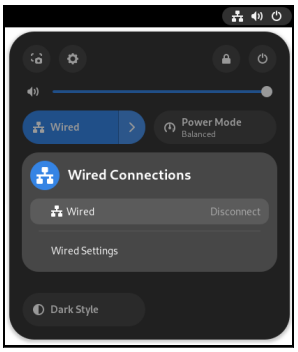


Figure 15.1 GNOME System Menu Displaying LAN and Wi-Fi Connections of NetworkManager

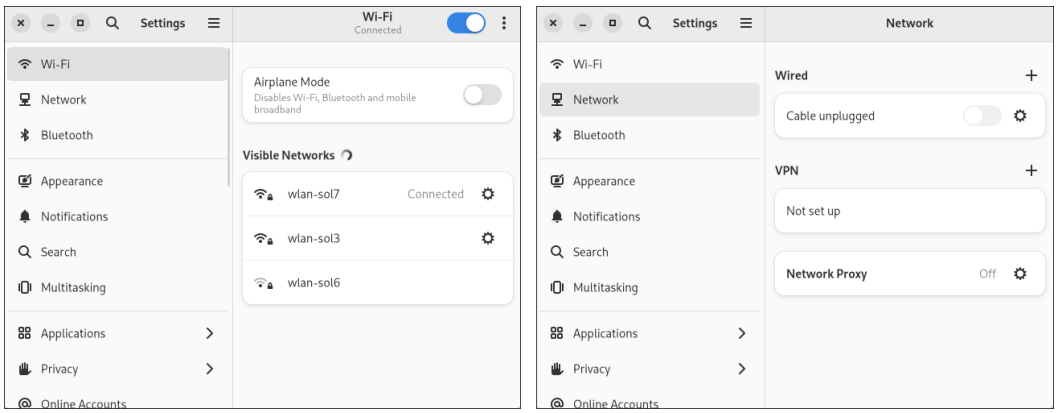


Figure 15.2 NetworkManager Settings Distributed across Network and Wi-Fi Modules of GNOME System Settings

Local networks with DHCP

The simplest use case for NetworkManager is when your computer is connected to a local network or router via a network cable. By default, NetworkManager checks for all LAN interfaces whether configuration information can be obtained via DHCP. If this succeeds, the network configuration already takes place fully automatically during the computer startup phase.

Explicitly Activating Network Interfaces

In the past, I had to deal with distributions where the automatic connection over an Ethernet cable failed. This happens pretty often with RHEL and its clones, where the installer includes an easily overlooked option that lets you specify whether you want to automatically establish a network connection.

If this option was set incorrectly during installation, the configuration can be easily corrected later: open the **Network** module of the system settings, click the gear icon, and then enable the **Automatically Connect** option.

Static LAN configuration

Static configuration of the LAN connection is required if your computer isn't connected to a router or DHCP server and you have to specify the IP address, netmask, gateway address, and name server address yourself (see Figure 15.3).

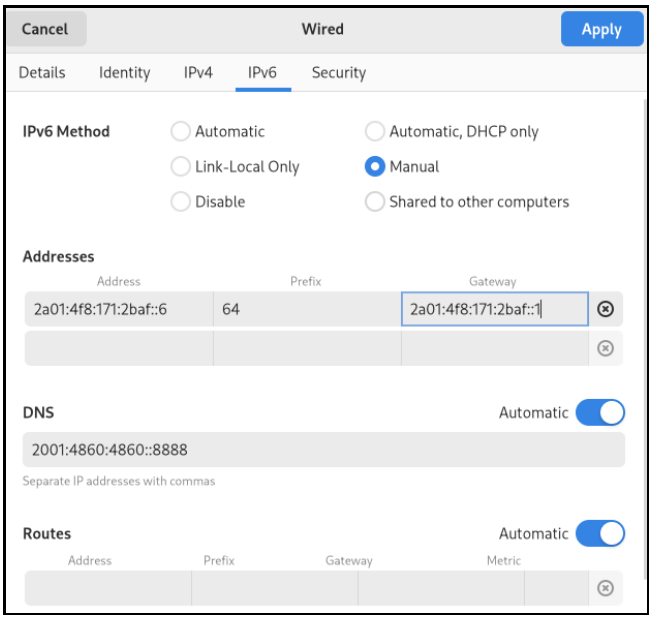


Figure 15.3 Static IPv6 Configuration of Linux Installation in Virtual Machine

To perform the configuration, you need to execute the **Network Settings** command in the NetworkManager menu. In the settings dialog, select the

Wired dialog sheet, then select the interface there, and use the gear icon to change its IPv4 and IPv6 settings. For static configuration, you must set the **IPv6 Method** field to **Manual**.

Setting up access to a wireless network

NetworkManager automatically detects all wireless networks within range. When you connect to a wireless network for the first time, you must specify the relevant password.

In corporate networks with the WPA and WPA2-Enterprise encryption systems, you must select a CA certificate in the NetworkManager settings dialog or select the **No CA Certificate Is Required** option (see Figure 15.4). Suitable certificates are located in the `/etc/ssl/certs` directory. The required certificate depends on the configuration of the company network, which is why you should ask the company administrator if necessary. Often the connection works even if you don't select a certificate at all.

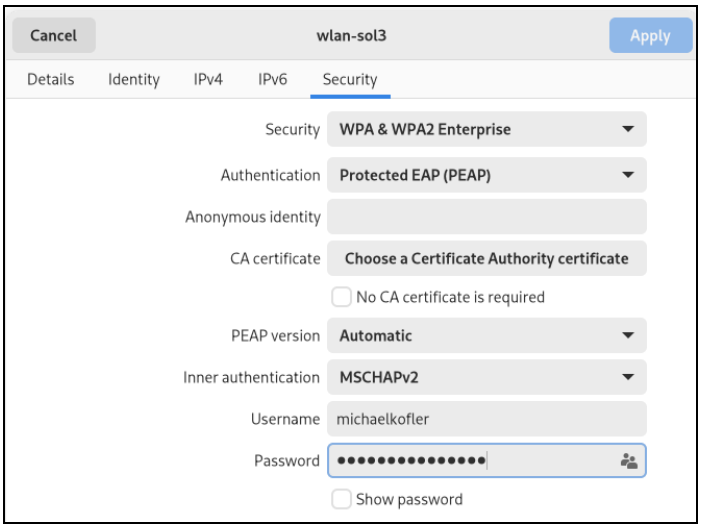


Figure 15.4 Wi-Fi Connection to Corporate Network

From then on, NetworkManager will then establish the connection independently. For this purpose, all Wi-Fi passwords are stored centrally, whereby the storage location varies depending on the distribution. On Fedora, NetworkManager sets up a text file in `/etc/sysconfig/network-scripts` for each wireless network connection. On Ubuntu, however, the Wi-Fi passwords are stored in the `/etc/NetworkManager/system-connections/*` directory.

A wireless network can be configured so that it doesn't broadcast its name. In this case it won't be displayed in the NetworkManager menu. To connect anyway, click **Connect to a Hidden Wireless Network** in the settings dialog.

Then you can specify the network name (ESSID = *Extended Service Set Identification*) and the encryption technique itself.

Setting up your own hotspot

If your computer obtains internet access via a LAN interface, you can use NetworkManager to configure the Wi-Fi controller so that your computer then acts as a hotspot and provides internet access to other wireless network devices within radio range. For smartphones, this procedure is often referred to as *tethering*; in network technology, it is more commonly referred to as a Wi-Fi access point.

The configuration is simple: In the Wi-Fi dialog, open the configuration menu via the **...** button and select the **Switch on Wi-Fi HotSpot** item. In a small dialog, you can then set the network name (more precisely the *Service Set Identifier* [SSID]) and a password. However, the hotspot function only works with Wi-Fi controllers whose Linux drivers support ad-hoc mode—which is by far not the case with all controllers.

15.1.2 Virtual Private Networks

You can use NetworkManager to perform a virtual private network (VPN) configuration based on an existing network connection. VPN is a technique to transfer network packets encrypted to another network. This enables, for example, secure access to a company network even from an insecure wireless hotel network.

NetworkManager can handle a relatively large number of VPN methods. However, depending on the distribution, you may need to install the package with the corresponding plugin first. The package names are `network-manager-<pluginname>` on Debian/Ubuntu, or `NetworkManager-<pluginname>` on Fedora and RHEL.

Unfortunately, the plugin name doesn't always indicate the VPN method. A good overview of the protocols that are supported and what the plugins are called can be found at *https://wiki.gnome.org/Projects/NetworkManager/VPN*.

You can set up a new VPN connection by clicking the plus button (**+**) in the **Network** dialog of the GNOME system settings and then selecting one of the supported methods (Cisco AnyConnect, OpenVPN, PPTP, etc.). This takes you to a dialog specific to the procedure in question. Don't be confused by the wealth of options; it's often sufficient to fill in just a few fields. Fields for the login name and password are sometimes not included in the dialog. NetworkManager only asks for this information when the connection is actually established.

15.1.3 Proxy Configuration

In some companies or organizations, HTTP traffic has to flow through proxy servers, even though this is not up to date in view of ubiquitous HTTPS connections. On GNOME, a gear icon in the **Network** dialog of the system settings takes you to the proxy configuration. You have the choice between an automatic or a manual configuration.

The proxy settings are stored in the GNOME-specific `dconf` database. You can also access the GNOME proxy settings using the following command:

```
user$ gsettings list-recursively org.gnome.system.proxy
...
org.gnome.system.proxy.http enabled true
org.gnome.system.proxy.http host '10.0.0.1'
org.gnome.system.proxy.http port 8080
```

After logging in to a GNOME session, you can set the respective environment variables such as `http_proxy` and `https_proxy`.

For the changed proxy configuration to take effect, you must log out and log in again.

Configure manually

The configuration options provided by KDE and some other desktop systems are comparable to those of GNOME. To configure the proxy settings separately from a desktop environment, the easiest way is to add the following lines to the `/etc/environment` file:

```
# file /etc/environment
http_proxy=http://10.0.0.1:8080
https_proxy=http://10.0.0.1:8443
```

More details about the manual proxy configuration can be found on the following web pages:

- *https://wiki.archlinux.org/title/Proxy_settings*
- *https://wiki.gnome.org/Projects/NetworkManager/Proxies*

Web browsers

All common web browsers should automatically adopt the proxy configuration. If necessary, you can change the proxy configuration directly in the settings dialogs of Firefox or Chrome.

Package management

To make sure that the package management tools also take the proxy into account, their configuration files may need to be modified. On Debian and Ubuntu, you must set the `Acquire::http::Proxy` parameter in an `apt` configuration file:

```
# for example in /etc/apt/apt.conf.d/01-proxy.conf
Acquire::http::Proxy "http://hostname-or-ipaddress:8080";
```

On RHEL or Fedora, you must add the following lines to `/etc/dnf/dnf.conf`:

```
# in /etc/dnf/dnf.conf
proxy=http://hostname-or-ipaddress:8080
# only if necessary
proxy_username=name
proxy_password=pw
```

15.1.4 Configuring NetworkManager in Text Mode

Up to this point, I have assumed that you work on GNOME or another desktop system with NetworkManager-compatible configuration tools. When you configure a server or a virtual machine, these requirements are rarely met.

Instead of manually editing the configuration files, which are very distribution-specific, you can call the `nmtui` command. (The command name stands for *NetworkManager Text User Interface*). Even though the design of the dialogs is minimalistic (see Figure 15.5), the program offers extensive configuration options. Among other things, you can use these options to set up VPN connections, network bridges, and so on.



Figure 15.5 Static Configuration of Network Connection Using nmtui

15.1.5 Internal Details

NetworkManager at command level

You can also manage all NetworkManager settings using the `nmcli` command. This allows you to control NetworkManager via scripts or, in the case of server installations, without a graphical user interface. The following commands first list all interfaces known to NetworkManager and then disable the Wi-Fi interface, `wlp0s20f3`:

```
root# nmcli dev
nmcli dev
DEVICE             TYPE      STATE                   CONNECTION
lo                 loopback  connected (externally)  lo
enp0s31f6          Ethernet  unavailable             –
wlp0s20f3          wifi      connected               wlan-sol3
br-72ad8975ca21    bridge    connected (externally)  br-72ad8975ca21
docker0            bridge    connected (externally)  docker0
virbr0             bridge    connected (externally)  virbr0
...
root# nmcli dev disconnect wlp0s20f3
```

The `nmcli dev show` command provides pages of information on all connection parameters for all interfaces.

Nameserver (DNS)

In most distributions, you can determine the address of the active nameserver by looking at `/etc/resolv.conf` or extracting the information from `nmcli dev show`:

```
user$ cat /etc/resolv.conf
# Generated by NetworkManager
search fritz.box
nameserver 192.168.178.1
nameserver fd00::3e37:12ff:feb7:a1cb
nameserver 2001:871:264:daef:3e37:12ff:feb7:a1cb

user$ nmcli dev show | grep -i dns
  IP4.DNS[1]: 192.168.122.1
```

Some distributions (e.g., Fedora and Ubuntu) delegate the administration of the DNS record to the `systemd-resolvd` service, and `/etc/resolv.conf` then points to a local address. The actual DNS address can again be determined using `nmcli dev show` or `resolvectl`:

```
user$ cat /etc/resolv.conf      (Fedora, Ubuntu)
  # This file is managed by man:systemd-resolved(8). Do not edit.
  nameserver 127.0.0.53
user$ nmcli dev show | grep -i dns
```

```
  IP4.DNS[1]:   192.168.178.1
  IP6.DNS[1]:   fd00::9a9b:cbff:fe06:8399
user$ resolvectl status | grep -i 'dns server'
  Current DNS Server: 192.168.178.1
  DNS Servers:        192.168.178.1 fd00::9a9b:cbff:fe06:8399
```

Background information about the `/etc/resolv.conf` file follows in Section 15.3, while `systemd-resolved` is described in Section 15.4.

Configuration files

The basic configuration for the network manager can be found in the following files:

```
/var/run/NetworkManager/conf.d/*.conf          (lowest priority)
/usr/lib/NetworkManager/conf.d/*.conf
/etc/NetworkManager/conf.d/*.conf
/etc/NetworkManager/NetworkManager.conf        (highest priority)
```

An elementary problem for NetworkManager is that many Linux distributions in the past developed their own systems to manage and store the network configuration (see also Section 15.4). The challenge for the cross-distribution NetworkManager was to maintain compatibility with all these distributions, a challenge that could be solved by using plugins. You decide where and in which syntax network settings are stored, such as static IP addresses, Wi-Fi passwords, and so on. The active plugins are specified in `/etc/NetworkManager/NetworkManager.conf`. The choices include the following:

- `keyfile`
  NetworkManager's own syntax (applies by default); configuration files in `/etc/NetworkManager/system-connections/`
- `ifcfg-rh`
  Deprecated Fedora/RHEL syntax; configuration files in `/etc/sysconfig/network-scripts/`
- `ifupdown`
  Debian syntax (ENI format); configuration file in `/etc/network/interfaces`

In many distributions, the tendency goes in the direction of `keyfile`. For example, on Debian and Ubuntu, NetworkManager stores its own connection data in `/etc/NetworkManager/system-connections`. The `/etc/network/interfaces` file is analyzed only to determine which interfaces NetworkManager is supposed to ignore.

The big exception used to be Red Hat–like distributions, which relied on the `ifcfg-rh` plugin for a long time. However, RHEL 8 has put an end to that. The Red Hat syntax, which is around 20 years old, is now considered obsolete

(but is still accepted for compatibility reasons). The new standard for RHEL and its clones is also the `keyfile` syntax of NetworkManager.

Dispatcher

It's possible to automatically execute scripts when establishing or terminating a connection. These scripts must be set up in the `/etc/NetworkManager/dispatcher.d/` directory. Details and examples of this method can be found on the following web page: *https://www.linux-magazine.com/Issues/2020/239/Dispatcher-Scripts*.

15.2 Manual LAN and Wi-Fi Configuration

Typically, your LAN or Wi-Fi controller is automatically detected and initialized when your computer boots. This section describes how this process works behind the scenes or which steps are required to perform the initialization manually. This helps you to better understand network fundamentals and more confidently use the user interfaces of common configuration tools.

15.2.1 Activating the LAN Controller Manually

The network or LAN controller is usually a chip on the mainboard of your computer that provides Ethernet network functions. However, the controller can also be integrated into the CPU.

Hardware detection

The first step is to make sure that the correct kernel module is loaded for your network controller. Often the kernel manages this automatically. In this case, the `ip link set enp4s0 up` command is executed without an error message. If problems occur at this point, you must determine which network controller is in your computer and which kernel module is responsible for it. First pieces of information are provided by `lspci` in such cases:

```
root# lspci -k | grep -i -3 net
...
00:1f.6 Ethernet controller: Intel Corporation Ethernet
                             Connection (7) I219-V (rev 10)
    Subsystem: Lenovo Ethernet Connection (7) I219-V
    Kernel driver in use: e1000e
    Kernel modules: e1000e
```

So the notebook uses an Ethernet controller from Intel and a driver that is included in the kernel module `e1000e`:

```
root# modinfo e1000e
filename:     /lib/modules/.../kernel/drivers/net/Ethernet/intel/
e1000e/e1000e.ko
```

```
description:  Intel(R) PRO/1000 Network Driver
...
```

Using `lsmod` you can now check if the module has already been loaded. This will typically be the case; that is, Linux has already recognized the controller correctly during system startup. Only if this is not the case do you have to load the appropriate module via `modprobe`:

```
root# modprobe e1000e
```

`dmesg` shows whether errors occur when loading the module (which is not the case here). The *link is not ready* warning only says that the interface is currently not active due to the lack of configuration:

```
root# dmesg
...
e1000e: Intel(R) PRO/1000 Network Driver - 3.2.6-k
e1000e: Copyright(c) Intel Corporation.
..
e1000e 0000:00:1f.6 enp0s31f6: renamed from eth0
ADDRCONF(NETDEV_UP): enp0s31f6: link is not ready
...
```

Usually the module is loaded automatically during the initialization of the computer. If that doesn't work, you need to enter the mapping between the interface and the kernel module in the module configuration file:

```
# module configuration file /etc/modprobe.d/config.conf
alias enp0s31f6 e1000e
```

List of network interfaces

Network interfaces have different names depending on the distribution. Some distributions still use the interface names `eth0`, `eth1`, and so on, but more and more distributions use device names like `enp0s5`. A list of all network interfaces available on your computer is provided by the `ip link show` command:

```
root# ip link show
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue ...
2: enp0s31f6: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc ...
3: wlp0s20f3: <BROADCAST,MULTICAST> mtu 1500 qdisc mq state ...
```

Activating the interface

Then you want to activate the relevant network interface using `ip link set <adapter> up`:

```
root# ip link set enp0s31f6 up
```

Configuring the interface

To configure the network interface, you need to run the `ip addr add` command. `ip addr show enp0s31f6` then displays all known network interface information:

```
root# ip addr add 192.168.0.2/24 dev enp0s31f6
root# ip addr show enp0s31f6
2: enp0s31f6: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc ...
    link/ether 00:1c:42:85:09:a1 brd ff:ff:ff:ff:ff:ff
    inet 192.168.0.2/24 scope global enp0s31f6
       valid_lft forever preferred_lft forever
```

Now you can use `ping` to check whether you can contact other computers on the local network. The `-c 2` option causes exactly two `ping` packets to be sent:

```
root# ping -c 2 192.168.0.1
PING 192.168.0.1 (192.168.0.1) 56(84) bytes of data.
64 bytes from 192.168.0.1: icmp_seq=1 ttl=64 time=2.95 ms
64 bytes from 192.168.0.1: icmp_seq=2 ttl=64 time=0.169 ms

--- 192.168.0.1 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1002ms
rtt min/avg/max/mdev = 0.169/1.560/2.952/1.392 ms
```

Commands: ip versus ifconfig and route

In many books and on websites about Linux network configuration, the `ifconfig` and `route` commands are still described. But these commands are obsolete!

`ip` provides various additional functions that are missing in `ifconfig` and `route`. `ip` is specially optimized for the network functions of the Linux kernel. Experienced administrators should now get rid of old habits and switch to the new `ip` command, while Linux beginners should steer clear of `ifconfig` and `route` right from the start.

Nameserver configuration

`ping` currently only works if you enter the correct IP address. To allow you to specify a host name instead, `/etc/resolv.conf` must contain the IP address of a name server. The following example assumes that the local network has its own nameserver with IP address 192.168.0.1. However, the nameserver can also be external and provided by the internet provider. Details about this configuration file follow in Section 15.3.2:

```
# /etc/resolv.conf
nameserver 192.168.0.1
```

Default gateway

At this point, packets can only be sent within the local network.

To enable contact to the internet, the computer must know where to route such packets. For this purpose, you must specify the address of the internet gateway of your network by using `ip route`. The following example assumes that the IP address of the gateway is 192.168.0.1:

```
root# ip route add default via 192.168.0.1
```

Without any additional parameters, `ip route` returns the routing table. The gateway address is contained in the line that starts with `default`:

```
root# ip route
default via 10.211.55.1 dev enp0s31f6
192.168.0.0/
24 dev enp0s31f6  proto kernel  scope link  src 192.168.0.2
```

Now it should be possible to send packets to any address on the internet:

```
root# ping -c 2 google.com
PING google.com (172.217.22.78) 56(84) bytes of data.
64 bytes from ....1e100.net (172.217.22.78): icmp_seq=1 ttl=56 time=
25.5 ms
64 bytes from ....1e100.net (172.217.22.78): icmp_seq=2 ttl=56 time=
25.0 ms
--- google.com ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 2ms
rtt min/avg/max/mdev = 25.013/25.271/25.529/0.258 ms
```

Enabling the loopback interface

A manual configuration of the loopback interface is rarely required. If it is, however, you need to execute the following two commands:

```
root# ip link set lo up
root# ip addr add 127.0.0.1 dev lo
```

Deactivating the interface

Basically, you can use `ip addr del` or `ip route del` to undo previously performed address assignments or route definitions. You must specify exactly the same parameters as for the corresponding `ip addr add` command:

```
root# ip addr add 192.168.0.2/24 dev enp0s31f6
root# ip addr del 192.168.0.2/24 dev enp0s31f6
```

To entirely disable a network interface—that is, to delete all address and route data—you must run `ip addr flush`:

```
root# ip addr flush dev enp0s31f6
```

15.2.2 Retrieving DHCP Information

Retrieving DHCP information

If there is a DHCP server on the network, you can use it for configuration purposes. After enabling the interface using `ip link set enp4s0 up`, you need to run the `dhclient` command in most distributions:

```
root# dhclient -v enp0s31f6
...

Listening on LPF/enp4s0/00:11:25:32:4f:5d
Sending on   LPF/enp4s0/00:11:25:32:4f:5d
Sending on   Socket/fallback
DHCPDISCOVER on enp4s0 to 255.255.255.255 port 67 interval 3
DHCPOFFER from 192.168.0.1
DHCPREQUEST on enp4s0 to 255.255.255.255 port 67
DHCPACK from 192.168.0.1
bound to 192.168.0.15 -- renewal in 36624 seconds.
```

Ad Hoc Network Configuration

The `dhclient` command often provides the fastest way to establish ad hoc network access on a computer that has not yet been configured. This means you do not need the `ip` commands and the manual setting of `/etc/resolv.conf`. The only requirement is a DHCP server on the local network.

15.2.3 IPv6 Configuration

All the examples so far have referred to IPv4. However, you can of course also use the `ip` command to perform an IPv6 configuration. The following commands assume that you have the IPv6 subnet 2a01:4f8:161:107::/64 and can use the gateway address fe80::1. The `enp0s31f6` interface is assigned the address 2a01:4f8:161:107::2. To try this out, use `ping6` to send commands to a server that you know is configured for IPv6:

```
root# ip -6 addr add 2a01:4f8:161:107::2/64 dev enp0s31f6
root# ip -6 route add default via fe80::1 dev enp0s31f6
root# ping6 -n google.com
PING google.com(2a00:1450:4009:805::1002) 56 data bytes
64 bytes from 2a00:1450:4009:805::1002: icmp_seq=1 ttl=54 time=19.7 ms
64 bytes from 2a00:1450:4009:805::1002: icmp_seq=2 ttl=54 time=19.1 ms
<Ctrl>+<C>
```

Testing the IPv6 configuration

The `ping6` command just mentioned is best suited for initial IPv6 testing. If `ping6` returns the error message *connect*: *The network is not reachable*, then you can take a closer look at the IPv6 configuration using the `ip` command.

`ip addr show <interface>` provides detailed data on the network interface `<interface>` in the following listing:

```
root# ip addr show enp0s31f6
2: enp0s31f6: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 ...
  link/ether 00:1c:42:d0:29:34 brd ff:ff:ff:ff:ff:ff
  inet  10.0.0.13/24 brd 10.0.0.255 scope global enp0s31f6
  inet6 fe80::21c:42ff:fed0:2934/64 scope link
    valid_lft forever preferred_lft forever
root# ip -6 route | grep default
- no result --
```

The line beginning with `link/ether` specifies the MAC address of the interface. The line beginning with `inet` contains the IPv4 address including a mask in the short notation (`/n`) as well as the broadcast address. The lines beginning with `inet6` specify the IPv6 addresses. Note that there can be several addresses. A "correct" IPv6 configuration requires the interface to be connected to a global unicast address, which usually starts with the number 2.

An IPv6 address alone is not enough. Linux also needs to know where to route IPv6 packets. You can use `ip -6 route | grep default` to determine the default gateway for IPv6.

In the previous example, there is only an IPv4 configuration. The address specified in the `inet6` line is just a link local address that is set up automatically and allows for the communication within a local network (comparable to Zeroconf for IPv4; see Section 15.5). No IPv6 gateway has been set up.

The next example shows the results on a root server with IPv4 *and* IPv6 configuration. The `enp0s31f6` interface is assigned the unicast address 2a01:xxx::2 (`scope global` in the `ip` result), and fe80::1 is set up as the IPv6 gateway:

```
root# ip  addr show enp0s31f6
3: enp0s31f6: <BROADCAST,MULTICAST,UP,LOWER_
UP> mtu 1500 qdisc noqueue state UP
    link/ether 54:04:a6:f1:74:1f brd ff:ff:ff:ff:ff:ff
    inet 5.9.22.18 peer 5.9.22.1/
32 brd 5.9.22.31 scope global enp0s31f6
    inet6 2a01:4f8:161:107::2/64 scope global
       valid_lft forever preferred_lft forever
    inet6 fe80::5604:a6ff:fef1:741f/64 scope link
       valid_lft forever preferred_lft forever
root# ip -6 route | grep default
default via fe80::1 dev enp0s31f6  metric 1024
```

15.2.4 Manual Control of the Wi-Fi Controller

Hardware detection

The kernel module for the Wi-Fi controller is usually loaded automatically. In most distributions, the interface for the wireless network is named `wlan0` or `wlpxxx`. If you're unsure, you can use `ip link` to determine a list of all interfaces. `dmesg | egrep -i "wlan|wifi"` returns all relevant kernel messages.

Determining the current status

Using `iw dev NAME info` and `iw dev NAME link`, you can determine the status of the interface. In the following example, an active connection exists:

```
root# iw dev wlp2s0 info
Interface wlp2s0
        ifindex 3
        type managed
        wiphy 2
root# iw dev wlp2s0 link
Connected to 00:16:b6:9d:ff:4b (on wlp2s0)
        SSID: wlan-sol2
        freq: 2462
        RX: 102262 bytes (1784 packets)
        TX: 12815 bytes (86 packets)
        signal: -59 dBm
        tx bitrate: 54.0 MBit/s
        ...
```

The pseudo file `/proc/net/wireless` also summarizes information about the current quality of the Wi-Fi connection:

```
root# cat /proc/net/wireless
Inter-| sta-|   Quality        |   Discarded packets               | Missed
| WE
 face | tus | link level noise |  nwid  crypt   frag  retry   misc | beacon
| 22
wlp2s0: 0020   91.  189.    0.       0      0      0      0      4        0
```

Setting up a Wi-Fi interface

If `iw` doesn't recognize the `wlp2s0` interface (error message *no such device*), you must set up the interface first. The `iw` command `interface add` is used for this purpose:

```
root# iw phy phy0 interface add wlp2s0 type managed
```

`phy0` addresses the first (and for most computers also the only) Wi-Fi controller. Instead of `managed`, you can also specify the following interface types: `monitor`, `wds`, `mesh` or `mp`, and `ibss` or `adhoc`. The new interface must then be activated:

```
root# ip link set wlp2s0 up
```

If you no longer need the interface, you can delete it again using `iw del`:

```
root# iw dev wlp2s0 del
```

Establishing a Wi-Fi connection

`iw scan` provides detailed information about all wireless networks within range. `grep` turns this into a simple list of all network names (SSIDs):

```
root# iw dev wlp2s0 scan | grep SSID
    SSID: myhome
    SSID: wlan-sol2
    ...
```

You can establish a manual connection to unsecured wireless networks using `iw ... connect`. You must specify the name of the network (i.e., the SSID). `iw` only takes care of the Wi-Fi connection. You then perform the Ethernet configuration using `dhclient` (`dhcpcd` on SUSE):

```
root# iw dev wlp2s0 connect hotel-wlan-1
root# dhclient wlp2s0
```

`iw disconnect` terminates the connection:

```
root# iw dev wlp2s0 disconnect
```

15.2.5 Encrypting the Wireless Network

Wi-Fi connection using WEP

If the wireless network is secured by the now completely insecure WEP method, you can pass the key with an additional parameter in the format `n:xxx`, where `n` indicates the key number (0 to 3). The actual key is specified optionally in the form of 5 or 13 ASCII characters or by 10 or 26 hexadecimal digits:

```
root# iw dev wlp2s0 connect hotel-wlan-1 keys 0:1a790bcc31
root# dhclient wlp2s0
```

Wi-Fi connection using WPA or WPA2

The procedure is somewhat more complicated if WPA or WPA2 are involved. In this case, the `wpa_supplicant` background program from the package of the same name is responsible for initializing the connection and for the further exchange of ever-changing keys. After its installation, you need to set up a configuration file, choosing a filename such as `/etc/wpa_supplicant.conf`.

The file may contain some global settings. This is followed by specific parameters for different wireless networks. The following example shows a minimum variant that is sufficient for establishing a connection to a Wi-Fi router or Wi-Fi access point using WPA or WPA2 personal encryption. The

two crucial parameters are `ssid` to identify the network and `psk` with the key encrypted again for security reasons (it is also permissible to specify the WPA key in quotes as plain text):

```
# /etc/wpa_supplicant.conf
network={
     ssid="sol"
     psk=
00a38f42e6681596e1a5a4c5ede9a15250fb2a01c21028c6d490bb3458b8ea00
}
network={
     ssid="wlan-sol2"
     psk=
053633deb59038da9e9168e015fef97d3d54ae3794d4a12d31ee75a830cccec2
}
```

When encrypting your WPA password, `wpa_passphrase` can help you. If you do not pass the password as a parameter, the command expects the password at the standard input. You can copy the result of this command directly into `wpa_supplicant.conf`, removing the line containing the password in plain text if possible:

```
root# wpa_passphrase wlan-sol2 'My very secret password!'
network={
        ssid="wlan-sol2"
        #psk="My very secret password!"
        psk=020d93e2ddb2cdee51e800b977ff7d58fde47d0913 \
        cd394f2133648a147f513f
}
```

Now you can start `wpa_supplicant`. The command runs until you end it by pressing Ctrl+C. It takes care of initializing the Wi-Fi connection and the subsequently regular renewal of the keys for the connection. In other words, the program must run as long as you use the Wi-Fi connection, so you should continue working in another console.

Let me make some brief remarks about the options of the command: `-i` specifies the network interface and `-c` the configuration file whose name you are free to choose. `-D` specifies the Wi-Fi driver(s) you are using. Just try `nl80211,wext`: this supports both the old Linux Wi-Fi drivers and the new implementations. `man wpa_supplicant` provides a list of all supported drivers:

```
root# wpa_supplicant -iwlp2s0 -Dnl80211,wext -c /etc/wpa_supplicant.conf
Trying to associate with 00:13:46:b5:25:6e (SSID='sol' freq=0 MHz)
Associated with 00:13:46:b5:25:6e
```

```
WPA: Key negotiation completed with 00:13:46:b5:25:6e [PTK=TKIP GTK=TKIP]
CTRL-EVENT-CONNECTED - Connection to 00:13:46:b5:25:6e completed (auth)
  [id=0 id_str=]
...
```

Links

For more tips on how to use `wpasupplicant`, visit the following page: *https://w1.fi/wpa_supplicant*.

iNet Wireless Daemon

There are considerations to replace `wpasupplicant` with a more modern system in the longer term. Whether and when the *iNet Wireless Daemon* (`iwd`) will succeed in doing so remains to be seen. For more information, visit the following pages:

- *https://iwd.wiki.kernel.org*
- *https://lwn.net/Articles/770991*
- *https://wiki.archlinux.org/title/Iwd*

15.3 LAN Configuration Files

This section introduces the most important configuration files for connecting the computer to a local network. Unfortunately, only some of these files have rules that are consistent across all major distributions. For the rest, the information compiled in this section refers to Debian, Fedora, openSUSE, RHEL, and Ubuntu (as of spring 2024). Typically, you will avoid directly modifying the configuration files and use your distribution's configuration tools instead.

For the following examples, the computer to be configured is called `uranus`. It is located on a local network with the domain named `sol`. Other computers on the local network are called `jupiter`, `saturn`, and so on. The local network uses 192.168.0.* addresses. The IP address of the local computer is 192.168.0.2. The IP address of the gateway computer on the local network is 192.168.0.1. The gateway computer runs its own nameserver. Names and numbers are of course only examples.

15.3.1 Basic Configuration

/etc/hosts

The `/etc/hosts` file contains a list of known IP addresses and their associated names. The file must contain the loopback interface data in any case. The minimal variant looks like this:

```
# /etc/hosts (minimal variant)
127.0.0.1   localhost
```

In most Linux distributions, `localhost` is also defined for IPv6. The following lines show the default settings on Fedora and RHEL:

```
# /etc/hosts (default configuration on Red Hat and Fedora)
127.0.0.1   localhost localhost.localdomain localhost4
localhost4.localdomain4
::1         localhost localhost.localdomain localhost6
localhost6.localdomain6
```

In the case of a static network configuration, such as on a root server, `hosts` can also contain an entry with the IP address and the host name of the computer:

```
# /etc/hosts (continued, static configuration of the local host)
...
192.168.0.2  uranus.sol uranus
```

If you want to address the other hosts on the local network by name and there is no local nameserver, you must also specify their names in `/etc/hosts`. So instead of `ping 192.168.0.3`, you can then simply run `ping saturn` to test the connection to the `saturn` computer:

```
# /etc/hosts (continued, static configuration of other hosts)
...
192.168.0.1   mars.sol    mars
192.168.0.2   uranus.sol  uranus
192.168.0.3   saturn.sol  saturn
```

Analogous entries are required in the `/etc/hosts` files of all computers on the local network. The more computers there are on the local network, the more tedious the administration of the `/etc/hosts` file becomes. For this reason, it is recommended to set up a nameserver on every network with more than two devices. Usually, this task is performed by the router on the local network. The device or the nameserver running in it then knows what all other computers on the network are called. The hosts on the local network can contact the nameserver to determine this information. `/etc/hosts` then only needs the `localhost` lines.

/etc/host.conf

The `/etc/host.conf` file specifies how TCP/IP should determine unknown IP addresses. The following sample file specifies that the `/etc/hosts` file should be analyzed first (`hosts` keyword), followed by querying the name server specified in `/etc/resolv.conf` (`bind`). The second line allows multiple IP addresses to be assigned to a host name specified in `/etc/hosts`. This file is present in almost all distributions in the form specified here and does not need to be changed (the `order` line applies only to old versions of the C library and can be omitted):

```
# /etc/host.conf
order hosts, bind
multi on
```

Gateway

On local networks, the address of the gateway is usually transmitted via DHCP.

For a static configuration, different files are responsible depending on the distribution. Many distributions use NetworkManager and store the configuration files in the default format (`keyfile`). In this case, the gateway address directly follows the IP address at the `address1` keyword:

```
# file /etc/NetworkManager/system-connections/<ifname>.conf
...
[ipv4]
# address 192.168.0.2 and gateway 192.168.0.1
address1=192.168.0.2/24,192.168.0.1
```

On Debian, `/etc/network/interfaces` describes all network interfaces. For statically configured interfaces, the gateway is set by the `gateway` keyword:

```
# in /etc/network/interfaces (Debian)
...
iface enp0s31f6 inet static
  address 192.168.0.2
  netmask 255.255.255.0
  gateway 192.168.0.1
```

15.3.2 DNS Configuration (resolv.conf)

/etc/resolv.conf

The `/etc/resolv.conf` file controls how IP addresses are determined for unknown network names (host names). *Unknown* here means that the names are not defined in `/etc/hosts`.

The `domain` and `search` keywords are used to expand incomplete names with the domain name—for example, `jupiter` to `jupiter.sol`. This primarily increases convenience because local hostnames can be specified in a shortened form. You can specify multiple domain names with `search`, but only one with `domain`. But note that the `domain` name has priority over the `search` names, which means it's tested first. If, as here, only a single domain name is specified, then the `domain` line can be omitted.

The most important entries in the `/etc/resolv.conf` file are prefixed with the `nameserver` keyword: this allows for up to three IP addresses of nameservers to be specified. These servers are always addressed when the IP address of an unknown computer name is to be determined. The specification of a

nameserver is absolutely necessary so that internet addresses can be resolved to IP addresses. Most routers run a local DNS, which in turn uses the DNS of the provider. Larger local networks usually have their own nameservers:

```
# /etc/resolv.conf
domain sol                # host names apply to .sol
search sol                # host names apply to .sol
nameserver 182.92.138.35  # first DNS
nameserver 195.3.96.67    # second DNS (if the first one fails)
```

If you use IPv6, you must also specify the address of an IPv6 nameserver. The following example provides the address of Google's public IPv6 nameserver:

```
# /etc/resolv.conf
...
nameserver 2001:4860:4860::8888
```

Automatism

Note that `resolv.conf` is mostly created dynamically:

- If the network connection (whether via LAN or Wi-Fi) is configured with DHCP, the script for establishing the connection or NetworkManager enters the nameserver addresses transmitted by the DHCP server.
- On Fedora and Ubuntu, the `systemd-resolved` service takes care of `resolv.conf`, whereby a local nameserver is interposed. Details of the procedure, which may be implemented in other distributions in the future, follow in Section 15.4. The actual (external) DNS server can be determined using the following command:

  ```
  user$ resolvectl status | grep -i 'dns server'
  ```

  You can get more background information on the concept of the local DNS resolver at the following website, for example: *https://blueprints.launchpad.net/ubuntu/+spec/foundations-y-local-resolver*.

15.3.3 Host Name

The current host name can be determined using the `hostname` command. Unless the host name is set by DHCP, the configuration is done in the `/etc/hostname` file for almost all current distributions. To change the host name, it's best to use the following command:

```
root# hostnamectl set-hostname my-new-hostname
```

Note that host names must not contain spaces or special characters. The - character is allowed, but not as the first or last character.

IDN

Umlauts and other special characters are actually not provided for in the host name. For internationalized domain names (IDNs), the host name is converted to ASCII characters according to the Punycode method. On Linux, the `idn` command helps. It's located in the `idn` or `libidn` package, depending on the distribution. The host name is the resulting string that starts with `xn--`. Only in the web browser of the clients does it turn into the "correct" representation:

```
user$ idn hellö-world.com
xn--hell-world-hcb.com
root# hostnamectl set-hostname xn--hell-world-hcb.com
```

15.3.4 Mappings between Controllers and Network Interfaces

With multiple network interfaces, it's often difficult to determine the mapping between the `ethN` or `enpNsM` devices and the physical hardware. Depending on the hardware, it's possible to make an LED integrated in the socket blink for 10 seconds using `ethtool -p enp0s31f6 10`. If the network driver doesn't support this operation, you will receive the *operation not supported* error message.

Naming rules

Current Linux distributions name network interfaces so that the device names of existing interfaces do not change even when additional network hardware is added.

The naming of devices is mostly done by systemd in combination with `udev` rules. Onboard devices are named `enoN`, PCIe adapters `ensN`, external devices `enpNsM`, and Wi-Fi adapters are assigned the name `wlpNsM`. Here, `N` and `M` each refer to hardware properties, such as the PCI slot. For more information, visit *https://freedesktop.org/wiki/Software/systemd/PredictableNetworkInterfaceNames*.

15.4 Distribution-Specific Configuration Files

The simplest case—the integration of a notebook computer into the local network (DHCP) or into the wireless network—is now left to NetworkManager by almost all distributions. (An exception is Raspberry Pi OS.) However, when it comes to special configuration variants for server use, various manufacturer-specific special paths have emerged over the last two decades.

Having introduced you to the most important configuration files in the previous section, which are the same in almost all distributions, I'll present

here some details by which the distributions differ from one another. In doing so, I go into detail about the following distributions:

- **RHEL and Fedora**
  `/etc/sysconfig/network-scripts` (deprecated)
- **Debian**
  `/etc/network/interfaces` (ENI)
- **SUSE**
  YaST and Wicked
- **Ubuntu**
  Netplan and `networkd`

The information compiled in this section is especially relevant if you want to perform a server configuration or use a Raspberry Pi as a Wi-Fi router or for similar functions. For the most part, no graphical user interface is available for configuration work. Also, you often have to do without the support of a DHCP server.

When Two Quarrel...

In many distributions, NetworkManager and the distribution's own configuration procedure run in parallel. Although NetworkManager tries not to touch interfaces that are explicitly managed by a distribution-specific procedure, sometimes this automatism fails and conflicts occur.

In the past, I would recommend simply disabling NetworkManager on servers (`systemctl disable NetworkManager`) or uninstalling it altogether. At least with Fedora and with RHEL from version 8 on, this is definitely not a good idea anymore! That's because NetworkManager has replaced all of its own network scripts there.

15.4.1 RHEL and Fedora (NetworkManager)

Deprecated: ifcfg

On Fedora and on RHEL from version 8 on, NetworkManager alone controls all network connections. Up to RHEL 7, various scripts and configuration files (`ifcfg-<interfacename>`) in the `/etc/sysconfig/network-scripts` directory were used instead.

There is still the compatibility plugin `ifcfg-rh` for NetworkManager, which can process files stored in the `network-scripts` directory. But in the meantime, it's high time to say goodbye to these files. For this reason, I've decided for the first time in this edition not to describe the old configuration system. If you really want or need to continue doing network configuration in this outdated form, I refer you to the RHEL 7 documentation at

https://access.redhat.com/documentation/de-de/red_hat_enterprise_linux/7/html/networking_guide/sec-configuring_ip_networking_with_ifcg_files.

New: keyfile

Fedora and RHEL now rely entirely on NetworkManager—even for server configuration. The configuration files for the interfaces use the default format of NetworkManager (the `keyfile` syntax) and are stored in the `/etc/NetworkManager/system-connections` directory. The syntax of these files is documented at the following pages:

- *https://www.linux.org/docs/man5/nm-settings-keyfile.html*
- *https://developer-old.gnome.org/NetworkManager/stable/nm-settings-keyfile.html*

In the following sections, I will show you some examples of typical configuration variants. However, I recommend that you perform the configuration in the NetworkManager dialogs. On a server without a graphical user interface, you can use the `nmtui` command from the `NetworkManager-tui` package instead (see also Figure 15.5). A direct modification of the configuration files in an editor should only be necessary in an absolute emergency.

Important Prerequisites

For NetworkManager to take your configuration files into account, they must be owned by `root` and must be readable only by `root`. Make sure to execute `chmod 600 filename`, and if necessary, also `chown root:root filename`!

During initial experiments with one's own configuration files, dynamically generated files in the `/var/run/NetworkManager` directory often interfere. Perform a reboot! NetworkManager creates dynamic configuration files only for those interfaces for which there is no explicit configuration in `/etc/NetworkManager`.

DHCP configuration

An Ethernet interface that receives its network configuration via DHCP does not actually need a configuration file. NetworkManager automatically takes care of the configuration. However, you can still write the configuration in a file like the following sample:

```
# file /etc/NetworkManager/system-connections/enp1s0.nmconnection
[connection]
id=myInterfaceName
uuid=27afa607-ee36-43f0-b8c3-9d245cdc4bb3
type=802-3-Ethernet
autoconnect=true
interface-name=enp1s0
```

```
[ipv4]
method=auto

[802-3-Ethernet]
mac-address=52:54:00:64:9b:85
```

`method=auto` is responsible for DHCP being used. You can create the mapping to the interface either via `interface-name` or via `mac-address`. The required hexadecimal address can be determined using `ip addr`. `autoconnect=true` causes the network interface to be activated automatically when the computer boots.

`id` is a freely selectable name for the connection. This name must be used with some `nmcli` commands to select the interface. It's often convenient to simply use the interface name (here `enp1s0`) as the ID.

Basically, you're also free to decide on the file name for the configuration file. However, it makes sense to follow the *<interface name>.nmconnection* syntax that applies on Fedora and RHEL. NetworkManager takes into account all files that are not clearly identifiable as backup files.

It takes a little persuasion to make changes to the configuration effective. You must first prompt NetworkManager to reread existing configuration files. Then you explicitly tell it your wish to actually apply the changes for a specific interface—here, `enp1s0`:

```
root# nmcli connection reload      (reload configuration)
root# nmcli dev reapply enp1s0     (apply new configuration)
root# nmcli connection show        (interface overview)
```

Static configuration

For a static IPv4 configuration, you need to change a few lines in the `[ipv4]` section:

```
# file /etc/NetworkManager/system-connections/enp1s0.nmconnection
...
[ipv4]
address1=192.168.122.244/24,192.168.122.1
dns=192.168.122.1
method=manual
```

`method=manual` activates the static configuration. `address1=...` assigns an IP address, the mask, and (if required) the corresponding gateway address to the interface. `dns` sets the nameserver you want. If required, you can also specify multiple nameserver addresses separated by commas.

The relevant lines for a static IPv6 configuration look as follows: **IPv6 configuration**

```
# file /etc/NetworkManager/system-connections/enp1s0.nmconnection
...
[ipv6]
addr-gen-mode=default
address1=2abc:1234:1234::10/64,2abc:1234:1234::1
dns=2001:4860:4860::8888,2001:4860:4860::8844
method=manual
```

In the `[ipv6]` section, `address1` again assigns the IP address including mask as well as the corresponding gateway address. The public Google nameservers are used as nameservers in this example.

You can include the desired zone for the `firewalld` program in the `[connection]` section of the configuration files: **Firewall zones**

```
# file /etc/NetworkManager/system-connections/enp1s0.nmconnection
[connection]
...
zone=trusted
```

Without this specification, the firewall system automatically uses the default `public` zone. The currently valid zones for all interfaces can be determined by using the `firewall-cmd --get-active-zones` command. Background information on firewall configuration follows in Chapter 29.

For an example of a network bridge configuration, please see Chapter 32, Section 32.4. **Network bridges**

15.4.2 Debian

On Debian, the `/etc/network/interfaces` file is responsible for configuring all interfaces. The configuration procedure is sometimes called *ENI*, derived from the first letters of the directories or files. **/etc/network/interfaces**

The syntax is clearer than for all other variants described in this book: each interface that is to be activated at computer startup must be named by `auto name`. `iface name options` describes the basic configuration of the interface. In case of a static configuration, the `address`, `netmask`, and `gateway` parameters follow in the other lines.

For the interaction with NetworkManager, it reads the `interfaces` file by default and only takes care of interfaces that are not named in `interfaces`.

If the computer is supposed to obtain the network parameters via DHCP, the entire configuration comprises only four lines! The first two lines **DHCP configuration**

activate the loopback interface, which is always required as it is used for computer-internal network communication. The two other lines activate the `enp0s31f6` interface:

```
# /etc/network/interfaces
auto lo
iface lo inet loopback
# dynamic connection to a DHCP server,
# which transmits the key data of the internet access
auto enp0s31f6
iface enp0s31f6 inet dhcp
```

Static configuration

If the connection to the internet is configured as static, the `iface` line contains the `static` keyword. The network parameters are subsequently specified by several keywords, the meanings of which are self-explanatory:

```
# /etc/network/interfaces
auto lo
iface lo inet loopback

# static network configuration
auto enp0s31f6
iface enp0s31f6 inet static
  address         211.212.213.37
  netmask         255.255.255.224
  gateway         211.212.213.1
```

Nameserver

For a static configuration, you must specify the nameserver addresses directly in `/etc/resolv.conf`:

```
# /etc/resolv.conf (Debian)
nameserver 211.222.233.244    # first DNS
nameserver 212.223.234.245    # second DNS
```

IPv6 configuration

If you also want to use IPv6, you can simply define the interface in question a second time in `/etc/network/interfaces` using the `inet6` keyword. The `auto` keyword specifies that the IPv6 configuration takes into account the so-called router advertisement of the gateway or IPv6 router. This procedure enables a kind of self-configuration of IPv6 networks:

```
# /etc/network/interfaces (automatic IPv6 configuration)
...
auto enp0s31f6
iface enp0s31f6 inet dhcp
iface enp0s31f6 inet6 auto
```

If the IPv6 gateway uses a DHCPv6 server, the correct method is `dhcp`. If, in addition, the router address is to be configured via router advertisement, then the additional `accept_ra 1` option is required. This is the case, for example, if you use `dnsmasq` with the `enable-ra` option as the DHCP server:

```
# /etc/network/interfaces (DHCPv6 configuration)
...
auto enp0s31f6
iface enp0s31f6 inet dhcp
iface enp0s31f6 inet6 dhcp
  accept_ra 1
```

For a static configuration, `interfaces` must look as follows:

```
# /etc/network/interfaces (static IPv6 configuration)
...
auto enp0s31f6
iface enp0s31f6 inet static
  ... (IPv4 configuration as before)

iface enp0s31f6 inet6 static
  address 2a01:4f8:161:107::2
  netmask 64
  gateway fe80::1
```

Wi-Fi and WPA

Wi-Fi interfaces are basically configured in the same way as Ethernet interfaces. You can control the interaction with `wpa_supplicant` using `wpa` parameters. In the simplest case, the configuration may look as follows:

```
# /etc/network/interfaces (manual Wi-Fi configuration with WPA)
auto wlan0
iface wlan0 inet dhcp
  wpa-ssid "wlan-name"
  wpa-psk  "top-secret"
```

If you want to set more WPA parameters, it is usually better to do this in a separate file, the location of which you can specify using `wpa-conf`:

```
# /etc/network/interfaces (manual Wi-Fi configuration with WPA)
auto wlan0
iface wlan0 inet dhcp
  wpa-conf /etc/wpa.conf
```

The following example shows the syntax of `wpa_supplicant`. Instead of the password in plain text, an equivalent but not so easily readable hexadecimal code was specified using `wpa_passphrase` (see Section 15.3):

```
# file /etc/wpa.conf
network={
  ssid="wlan-name"
  psk=113df295cd91605ce30c00d16b82bf5eb334b0e47adcf9aafc153459a731fa42
  proto=RSN
  key_mgmt=WPA-PSK
  pairwise=CCMP
  auth_alg=OPEN
}
```

A lot of other `wpa` parameters for the `interfaces` file are documented in the following file:

/usr/share/doc/wpasupplicant/README.modes.gz

Running scripts You can use the `up` and `down` keywords to include commands in the `interfaces` file that are automatically executed when an interface is enabled or disabled.

Activating changes To activate a new configuration for an interface, you can run `ifdown` and `ifup` for that interface:

```
root# ifdown enp0s31f6
root# ifup enp0s31f6
```

These commands are part of the `ifupdown` package. The `ifup -a` command analyzes `/etc/network/interfaces` and activates all `auto` interfaces. If interfaces are configured via DHCP, `ifup` uses the `dhclient` command to transfer and analyze the DHCP data. The `/etc/dhcp/dhclient.conf` file is responsible for the configuration.

15.4.3 SUSE

YaST (SUSE) On SUSE, YaST decides on the default network configuration procedure during the installation. On notebook computers, NetworkManager is used, while on servers, SUSE's own development named `wicked` runs by default (see *https://github.com/openSUSE/wicked*). This daemon is particularly optimized for servers where network connections change frequently.

Of course, you can change the procedure later in the **System • Network Settings** YaST module, which you can find in the **Global Options** dialog sheet (see Figure 15.6). If you set **Wicked Service** there, YaST lists in the **Overview** dialog all network interfaces it has found. Then you can edit the individual interfaces. By default, YaST configures all Ethernet interfaces to obtain their IP address and all other configuration settings via DHCP from the router of

the local network. Note that YaST can also be started in text mode in minimal installations.

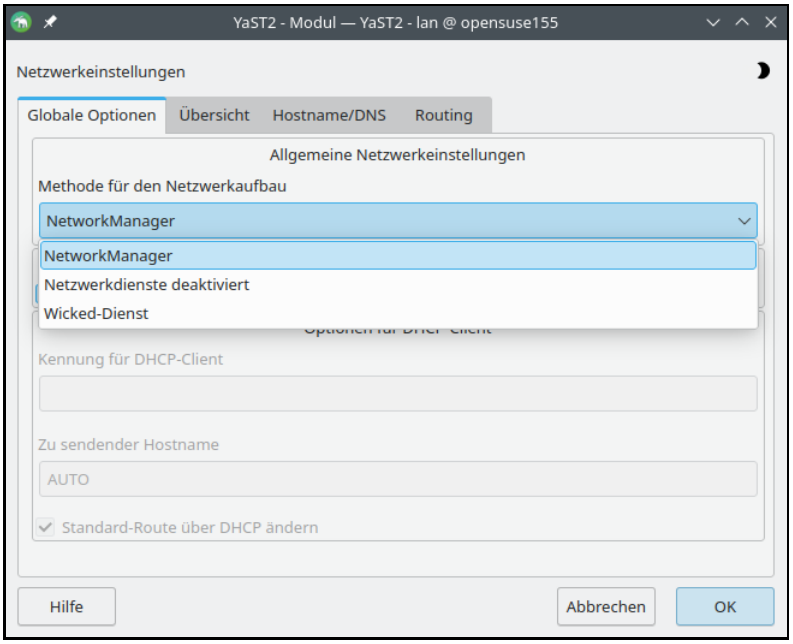


Figure 15.6 Setting Network Configuration Procedure Using YaST

Regardless of the configuration method you choose, you can set the hostname in YaST via **System • Network Settings • Hostname/DNS**. This is often necessary because the hostname specification is not provided during the SUSE installation. Instead, SUSE uses a randomly generated name that is correspondingly difficult to remember.

15.4.4 Ubuntu

On Ubuntu, the Debian-typical `/etc/network/interfaces` configuration file used to play a major role. That changed in 2017. Since then, the network configuration has been based on three pillars:

- The Netplan configuration system, a proprietary development by Ubuntu and Canonical, is the linchpin.
- For desktop installations, Netplan transfers the configuration to NetworkManager via all interfaces.
- On servers, on the other hand, Netplan interacts with `networkd`. This is a network configuration system that is part of systemd and hasn't been widely used yet.

Netplan

`netplan` is a network backend that communicates with NetworkManager on desktop systems and with `networkd` on server systems. The configuration is done by `*.yaml` files in `/etc/netplan`. On the following page, `netplan` is wonderfully documented with many real-life examples: *https://netplan.readthedocs.io/en/stable/examples*.

Netplan on desktop PCs

On notebook computers and desktop PCs, the following configuration file is sufficient for all network interfaces to be controlled by NetworkManager:

```
# file /etc/netplan/01-network-manager-all.yaml
# NetworkManager should take care of all network
# interfaces
network:
  version: 2
  renderer: NetworkManager
```

Netplan on the server

On Ubuntu Server, NetworkManager is not installed by default. This is why you have to deal with the `netplan` syntax. In virtual machines or cloud installations, `/etc/netplan` contains the following automatically generated configuration file. It causes the Ethernet interface (here `enp1s0`) to obtain its IP configuration via DHCP:

```
# file /etc/netplan/50-cloud-init.yaml
# This file is generated from information provided by the datasource.
Changes
# to it will not persist across an instance. To disable cloud-init's
network
# configuration capabilities, write a file
# /etc/cloud/cloud.cfg.d/99-disable-network-config.cfg with the
following:
# network: {config: disabled}
network:
    Ethernets:
        enp1s0:
            dhcp4: true
    version: 2
```

To perform a static configuration, you want to delete this file and then use the following listing as a guide:

```
# file /etc/netplan/10-static.yaml
network:
  Ethernets:
    enp1s0:
      addresses:
        - 192.168.122.201/24
```

```
    gateway4: 192.168.122.1
    nameservers:
      addresses:
        - 192.168.122.1
```

Changes to the configuration files take effect via `netplan apply`:

netplan command

```
user$ sudo netplan apply
```

`netplan` uses this to create temporary configuration files for `networkd` (see the following section) in the `/run/systemd/network` directory. A look at the files located there can help with any troubleshooting that may be required.

For an example of a network bridge configuration, please see Chapter 32, Section 32.4.

Network bridges with netplan

15.4.5 networkd (systemd)

The systemd developers have not only set themselves the task of modernizing the init process (they have succeeded!), but they also want to bring the configuration of different Linux systems down to a common denominator. Accordingly, systemd contains various modules for network configuration.

The `/etc/systemd/network` directory is intended as the location for the configuration files, while the `systemd-networkd` background process takes care of the analysis. The syntax of the configuration files seems well-thought-out and documented; you can look them it using `man systemd.network`. The wiki documentation of Arch Linux is also very useful (see *https://wiki.archlinux.org/title/systemd-networkd*).

The following lines show an example of the setup of an Ethernet interface with a static IP address:

```
# file /etc/systemd/network/static.network
[Match]
Name=enp2s0
[Network]
Address=10.0.0.15/24
Gateway=10.0.0.138
DNS=10.0.0.138
```

Although the infrastructure for systemd network configuration is already available as standard in many distributions, only a few distributions make use of these options. Currently, the most important exception is Ubuntu (see Section 15.4.4). Its `netplan` system creates temporary `netword` configuration files in the `/run/systemd/network` directory.

15.5 Zeroconf and Avahi

In this book, I generally assume that you either configure the computers in your network yourself or obtain the IP configuration from a central router or DHCP server. But there is a third way: automatic configuration by Zeroconf.

In this procedure, all computers connected to the network exchange their configuration data. New computers or devices connected to the network use this information to configure themselves so that they can communicate with the other devices without conflicts. The automatically configured hosts use addresses from IP range 169.254.*.* and host names ending in `.local`. Zeroconf communication takes place via UDP port 5454. For Zeroconf to work, this port must not be blocked by a firewall within the LAN!

Zeroconf was first implemented by Apple as *Rendezvous*. This project was later renamed *Bonjour* and is also available for Windows. This implementation is available as open-source code, but the license is not GPL-compatible. For this reason, a separate Zeroconf project was created for Linux under the name *Avahi*, the code of which is independent of Bonjour. The license used is LGPL.

Thanks to Zeroconf, computers, printers, Wi-Fi-enabled lamps, and other home automation devices can see each other without complicated configuration. If you want your Linux machine to be visible to Apple devices, the machine must be running the Avahi daemon. You can find more technical details at the following page: *https://avahi.org*.

Zeroconf Is Only Required for IPv4

Zeroconf is an IPv4-specific method. In IPv6, there is no need for Zeroconf because each network interface is automatically assigned a link-local IPv6 address. This address is needed for IPv6-internal configuration tasks, but it can also be used for Zeroconf-type applications.

avahi-daemon

The `avahi-daemon` service is responsible for the communication between Avahi computers. The configuration is carried out via `/etc/avahi/avahi-daemon.conf`, where you can usually retain the basic settings. The only exception is often the `enable-dbus` variable: it controls whether Avahi allows the communication method. Some Avahi-compatible programs require D-Bus. To enable D-Bus, you need to modify `avahi-daemon.conf` as follows:

```
# /etc/avahi/avahi-daemon.conf
[server]
...
enable-dbus=yes
```

Then restart the service:

```
root# systemctl restart avahi-daemon
```

Name resolution

If you want to address external hosts with ordinary, non-Avahi network programs via their `.local` names (e.g., via `ping merkur.local`), you need to install the `avahi-dnsconfd` package and start the eponymous network daemon:

```
root# apt/dnf install avahi-dnsconfd
root# systemctl start avahi-dnsconfd
root# systemctl enable avahi-dnsconfd
```

This is a kind of nameserver for Avahi host names. You do not need this daemon if you have a nameserver running on your network anyway.

To ensure that all programs use `avahi-dnsconfd` for name resolution, you must make sure that the `libnss-mdns` library is installed and that the `hosts:` line in `/etc/nsswitch.conf` contains the `mdns4` keyword. In some distributions, this is the case by default:

```
# in /etc/nsswitch.conf
...
hosts: files dns mdns4
```

Browsing

After this preparatory work, you can find out which computers or services are recognized by Avahi on your network. The console command `avahi-browse -a -t` or its graphical equivalent Avahi Discovery (program name `avahi-discover`) can help you in this regard. The commands must be installed separately in many distributions and can be found in the `avahi-tools` and `avahi-ui-tools` packages on Fedora, for example.

The GNOME file manager shows all Avahi machines in the network view by default. The KDE file manager provides a similar function under the `zeroconf:/` address, provided that the corresponding extension is installed (depending on the distribution, this could be included in the `kde-zeroconf` package). Various messaging and multimedia applications also support Zeroconf.

Chapter 16
Software and Package Management

This chapter describes how you can install and update software or packages on Linux and which techniques are used. Central topics of this chapter are the RPM and DEB package formats; the `apt`, `dnf`, `dpkg`, `pacman`, `rpm`, `yay`, and `zypper` package management commands; and the new AppImage, AppStream, Snap, and Flatpak package systems.

16.1 Introduction

On Windows, it's common to install new programs from a Microsoft software installer (MSI) file or by running `setup.exe`. The setup package contains all files required for the program.

Linux takes a completely different approach: a package management system maintains a database with information about all software packages already installed, and new programs are installed by the commands and downloaded from central package sources on the internet. This concept has a lot of advantages:

- Larger programs can be divided into independent packages with libraries and language files (localization). This reduces redundancy and makes it possible to install only those components that are really needed.
- Package management takes into account dependencies and conflicts between software packages. For example, if a program A requires library B, the package management system will not allow A to be installed until B has been installed.
- The package database can be used at any time to trace to which package a particular file belongs and whether this file is still in its original state.
- All packages installed on a computer can be updated together. This central update system is probably the biggest advantage compared to Microsoft Windows, where almost every major program maintains its own update program.

Two package management systems dominate the Linux market:

Package formats

- **RPM**
  Fedora, RHEL, SUSE and countless other distributions use the RPM package format developed by Red Hat.

- **DEB**
  Debian, Ubuntu, and all their derived distributions, on the other hand, use the DEB package format.

However, the commands for installing, uninstalling, and updating these packages (`rpm`, `dpkg`, and so on) are relatively primitive. You cannot download packages from package sources or resolve package dependencies.

Package management systems

Based on `rpm` or `dpkg`, there are package-management commands with many additional functions. This includes automatically installing dependent packages, performing updates for the entire system, and taking into account package sources from the internet. Examples of such package management systems include DNF, Yum, and ZYpp for RPM packages and APT for DEB packages.

Other formats

Besides the tools related to the RPM and DEB package formats, some distributions have developed their own package formats and tools, such as Arch Linux with Pacman and ClearLinux with `swupd`.

PackageKit

To enable distribution-independent tools and interfaces despite the diverse formats, the PackageKit library was developed. It provides a standardized interface to various package management backends (see Figure 16.1). As I will explain in Section 16.8, PackageKit doesn't seem to have a future despite its universal concept.

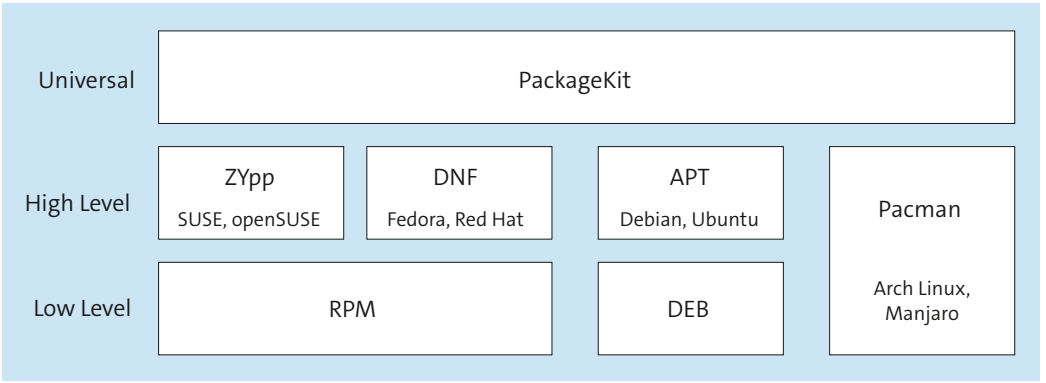


Figure 16.1 Low-Level and High-Level Package Management Systems

Distribution-specific tools

In addition to these standard programs, some distributions or desktop systems have their own programs for package management and for performing updates:

- **GNOME**
  Software (`gnome-software`)
- **KDE**
  Discover, as well as various other programs

- **Manjaro**
  Pacman Manager (`pamac`)
- **Ubuntu**
  Ubuntu Software, `update-manager`
- **SUSE**
  YaST modules of the **Software** group

Not only do the package management tools differ from distribution to distribution, there are also many other distribution-specific peculiarities despite common standards. These are summarized in Section 16.12.

Avoid Mixing Packages from Different Distributions

The packages of a Linux distribution are coordinated with each other. This means that they use unified libraries, were compiled by the same compiler, and so on. As a Linux beginner, you are therefore well advised to install only packages that are intended for your distribution. It is not recommended to install a Red Hat package on SUSE (or vice versa). The problems that arise, such as missing libraries or unfulfilled package dependencies, can only be fixed by Linux professionals, if at all.

Central administration

If you are responsible for 50, 100, or 1,000 Linux machines, the administration and package management becomes a pain despite the tools presented in this chapter. You need a tool to perform an update centrally on all or on previously selected computers, to install a new program, or to change the configuration. Depending on the distribution, *Red Hat Satellite*, *m23* (various distributions), or *Landscape* (Ubuntu) can be used. Visit the following pages:

- *https://m23.sourceforge.io*
- *https://landscape.canonical.com*
- *https://www.redhat.com/en/technologies/management/satellite*

16.1.1 Disadvantages of Linux Package Management

The package management of common distributions works well, but it also has its disadvantages:

- The almost daily updates of many distributions unsettle users who are used to longer update periods from Windows or macOS.
- The download requirement for the updates is large.

- Many Linux users want to use a distribution for a longer period of time while updating a few programs—often desktop applications like LibreOffice or GIMP. This is exactly what current distributions make virtually impossible. There are security updates available, but not fundamentally new versions.

If you want to use the latest version of LibreOffice in an older distribution, you must perform a manual installation or resort to nonofficial package sources. Both are recommended for advanced users only. On Windows or macOS, it's much easier to install the current LibreOffice version.

There are technical reasons why this is so: Most Linux programs use countless libraries. Therefore, a version update of LibreOffice implies that some libraries have to be updated as well. This in turn can trigger incompatibilities with other programs that also rely on this library.

A possible way out is to integrate the associated libraries for important programs. Google Chrome has taken this approach right from the start, and the Firefox and Thunderbird packages are now also maintained in this way. But this approach is also associated with disadvantages due to the redundancies that are now unavoidable, the space required on the hard disk, the download volume for each update, and the RAM requirement increase when you run multiple programs simultaneously. If a security problem occurs in a library, it can no longer be fixed centrally. Rather, all programs that use this library must be updated.

16.1.2 New Concepts

RPM and DEB packages will probably remain with us for a while as the basis for package management. At the same time, however, alternative concepts are increasingly becoming established. On the one hand, they are intended to supplement RPM/DEB and compensate for their weaknesses; on the other hand, they can completely replace conventional package management in special cases.

AppStream

AppStream is a method developed by Red Hat (originally referred to as *Modularity*) to offer different versions of a program in parallel in traditional DNF package sources. As a user, you then have the choice, for example, of installing PHP version 8.1 or 8.2.

AppStream is particularly well suited for long-lived server distributions with a stable, largely unchanging foundation to offer selected tools (server applications or programming languages) in current versions.

Flatpak and Snap

Flatpak, favored by Red Hat and Fedora, and *Snap*, developed by Canonical/Ubuntu, make it easier to offer current programs in current versions, independent of the distribution. Both package systems operate in parallel with conventional package management. From a user perspective, they facilitate the installation of (desktop) programs outside of the official package offerings. At the same time, a sandbox system mitigates potential security issues. A detailed description of these two package systems follows in Section 16.11.

In the long term, there is probably no way around these package systems. On Fedora and RHEL, LibreOffice will only be available via Flatpak in the future. On Ubuntu, Firefox can only be installed as a Snap package. As of version 23.10, this should also apply to the CUPS printing system.

Zero Install and AppImage

The *Zero Install* and *AppImage* projects pursue similar goals to Flatpak and Snap. AppImage in particular seems mature and has even received the blessing of the usually critical kernel developer Linus Torvalds: "This is just very cool." Without the support of major distributions, however, it is unlikely that these systems will catch on. See the following pages for more info:

- *https://0install.net*
- *https://appimage.org*

Silverblue, MicroOS, Vanilla OS

Finally, there are considerations to use the root file system in read-only mode most of the time. Updates create an updated file system in a second partition or in a `btrfs` snapshot. The active partition or branch is then changed during the subsequent boot process. (This is similar to how updates work on modern smartphones.) These ideas require a complete rebuild of both the package and boot systems. Some experimental distributions implement such concepts. For more information, visit the following pages:

- *https://vanillaos.org*
- *https://microos.opensuse.org*
- *https://silverblue.fedoraproject.org*
- *https://ubuntu.com/blog/ubuntu-core-an-immutable-linux-desktop*

16.2 RPM Package Management

The `rpm` command installs and manages RPM packages. It helps to do the following:

- Run scripts as part of an installation
- Ensure that all prerequisites are met before installing a program—that is, that all required libraries are available in the correct version

- Check if a file has been modified since the package was installed
- Determine to which package a particular file belongs
- Restore all files of a program during an uninstallation

The required management information is in each RPM package.

During the installation, this information is entered into a database whose files are located in the `/var/lib/rpm` directory.

16.2.1 Basic Principles

Most RPM packages are provided in two variants: as a binary package and as a source package. The binary package contains the files necessary to run the program. However, the source code package is only interesting for developers. It contains the source code that was required to assemble the binary package.

The package name contains quite a lot of information: for example, `abc-2.0.7-1.x86_64.rpm` denotes package `abc` with version number 2.0.7 and release number 1. If an error has occurred during the compilation of a package, additional online documentation has been added, or other changes have been made, release numbers greater than 1 are generated for a specific version number. So the version number refers to the actual program and the release number to the `rpm` compilation.

The `x86_64` identifier indicates that the package contains binaries for Intel/AMD-compatible 64-bit processors. If package `abc` contains script or text files that are independent of the CPU architecture, the `noarch` abbreviation is used instead of the CPU identifier. If the package contains the source code, the `src` abbreviation is commonly used instead.

Metadata

In addition to the files to be installed, the package file contains a lot of administrative information: a brief package description, information about version numbers, placement in the group hierarchy, dependencies on other packages, and so on. Dependencies exist when a package requires a particular programming language, such as Perl, or a particular library. In this case, these packages must be installed first. `rpm` maintains a database with information about all installed binary packages. This database is stored in various files in the `/var/lib/rpm` directory. The database contains only information about binary packages. Any installed packages with source code are not included in the database.

To make the RPM database match the actual installation, packages must be removed not by simply deleting the files, but by uninstalling them (`rpm -e`)!

Repairing the RPM Database

In rare cases, it happens that the RPM database contains inconsistent data. This manifests itself in the fact that the `rpm` command can no longer be used or returns error messages such as *cannot open packages database*. The following commands usually provide a solution:

```
root# rm -f /var/lib/rpm/__db*
root# db_verify /var/lib/rpm/Packages
root# rpm --rebuilddb
root# dnf clean all
```

This will recreate the RPM database. However, it takes a while.

Delta RPM packages

To update an RPM package, you usually download the entire new package. This is particularly inefficient for security updates, which often require only tiny changes to a few files. For this reason, there are delta RPM packages that contain only the changes from a particular version of the package.

When applying delta RPMs, the `applydeltarpm` command creates the new, updated RPM package from the delta RPM and the original package or its installed files. This is then installed as normal. `applydeltarpm` is part of the `deltarpm` package. The download volume is less valuable today than when this technology was introduced over 15 years ago because the use of delta packages is more complex than the simple installation of a complete package and because the parallel storage of ordinary packages and delta packages is redundant, so distributors are increasingly questioning this concept. Fedora in particular has concrete plans to do away with delta packages altogether in the near future, possibly starting with version 39 (see *https://lwn.net/Articles/925348*).

16.2.2 The rpm Command

It may seem surprising, but you will rarely use `rpm` to directly install a package or remove it again. For this purpose, you usually use `dnf`, `zipper`, or a graphical user interface.

The practical use of the `rpm` command today is primarily to read the package database and extract information from it that `dnf` or `zypper` won't give you at all, or only in a much more cumbersome way. Table 16.1 summarizes the most important `rpm` commands. The following examples show the practical application.

Task	Command
Installing a package	`rpm -i file.rpm`
Updating a package	`rpm -U file.rpm`
Verifying a package installation	`rpm -V file.rpm`
Removing a package	`rpm -e package name`
Determining all installed packages	`rpm -qa`
Determining the package that provides this file	`rpm -qf file`
Displaying a package description	`rpm -qi package name`
Determining a list of all files of the package	`rpm -ql package name`
Determining a list of all configuration files of the package	`rpm -qc package name`
Obtaining information about a package that is not yet installed	`rpm -qpli file.rpm`

Table 16.1 Important rpm Commands

Examples

Let's suppose you discover a file in the `/etc` directory that you have never noticed before and want to know what its purpose is. `rpm -qf` will tell you which package it belongs to, `rpm -qi` provides a brief description of the package, and `rpm -ql` shows all the other files that come from that package:

```
user$ rpm -qf /etc/login.defs
shadow-utils-4.13-6.fc38.x86_64
user$ rpm -qi shadow-utils
Name        : shadow
Summary     : Utilities for managing accounts and shadow password files
...
user$ rpm -ql shadow-utils
/etc/default/useradd
/etc/login.defs
/usr/bin/chage
...
```

You may want to know which `perl` packages are installed. `rpm -qa` provides a list of all installed packages. Use `grep` to filter out the relevant packages and `sort` to sort the list:

```
user$ rpm -qa | grep perl | sort
perl-AutoLoader-5.74-495.fc38.noarch
perl-B-1.83-495.fc38.x86_64
perl-base-2.27-495.fc38.noarch
...
```

Say that your Python installation is causing problems and you aren't sure whether the package has been installed correctly. Are all installed files of this package still in their original state? To find out, run `rpm -V`. This command lists all the files that have changed. Usually, the result should be empty, as in the following example, or contain only configuration files:

```
user$ rpm -V python3
```

On Fedora or RHEL, you can then repair the package with `dnf reinstall` if necessary:

```
root# dnf reinstall python3
```

16.3 DNF

DNF versus Yum

DNF (command name `dnf`) helps you to manage RPM packages on Fedora and RHEL. The original name of the command was Yum; it was only later reimplemented as DNF. For compatibility reasons, you can continue to use `yum` as the command name, but you will actually be running `dnf`. And so it happens that countless books and manuals refer to the no longer existing command `yum` to this day.

The Babylonian language confusion also includes the configuration files. DNF uses Yum repositories defined by files in `/etc/yum.repos.d`. On the other hand, there is also the configuration file `/etc/dnf/dnf.conf` with some global settings.

16.3.1 Concept

DNF simplifies the management of RPM packages. It provides numerous additional functions compared to `rpm`:

- Yum archives on the internet serve as the data sources (*repositories*). A repository is a collection of RPM packages for which additional `metadata` is stored in the `repodata` directory. They provide information about the contents and dependencies of all packages.
- DNF can manage several mirrors for a package source and tries to use the fastest mirror at the time.

- DNF resolves package dependencies, loads all required packages, and installs them. For example, if you install a package from package source A, DNF may download dependent packages from sources B and C and install them as well, after prompting you.
- DNF can update all already installed packages via a single command. For this purpose, each package is tested to see whether there is a newer version of the package in one of the registered package sources. If that is the case, the appropriate packages are downloaded and installed. Of course, all package dependencies are resolved in the process as well.

Locking conflicts

You cannot run multiple DNF instances in parallel. If a DNF command or program is already running, starting it again will result in an error message: *another copy is running*.

16.3.2 Configuration

The basic configuration of DNF is carried out by the `/etc/dnf/dnf.conf` file. The `/etc/yum.conf` link also references `dnf.conf`. The following lines show the configuration of RHEL 9:

```
# file /etc/dnf/dnf.conf (RHEL 9)
[main]
gpgcheck=1
installonly_limit=3
clean_requirements_on_remove=True
best=True
```

`gpgcheck=1` causes DNF to use a key to ensure the authenticity of packages. `gpgcheck` can also be set individually for each package source, different from the setting in `dnf.conf`. `plugins` decides whether DNF takes plugins into account.

There are packages that DNF is supposed to install but not update. These include kernel packages in particular. During a kernel update, the new kernel package is installed in parallel without affecting the old kernel package. The `installonlypkgs` variable is used to set the names of such packages. By default, this variable has the following setting: `kernel`, `kernel-smp`, `kernel-bigmem`, `kernel-enterprise`, `kernel-debug`, `kernel-unsupported`. Finally, the `installonly_limit` variable included in `dnf.conf` controls how many versions of such packages are installed in parallel. The default setting `3` causes only the latest three kernel versions to remain installed at all times. Older kernel packages are removed. `best=True` means that updates should install the latest available version, even if dependencies cannot be met then. `man dnf.conf` actually recommends the safer `false` setting here.

Package sources

Each package source is defined in its own `*.repo` file in the `/etc/yum.repos.d` directory. The following lines show the package source for Fedora's base packages:

```
# file /etc/yum.repos.d/fedora.repo (shortened)
[fedora]
name=Fedora $releasever - $basearch
metalink=https://mirrors.fedoraproject.org/ metalink?repo=fedora-
$releasever&arch=$basearch
enabled=1
metadata_expire=7d
repo_gpgcheck=0
type=rpm
gpgcheck=1
gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-fedora-$releasever-
$basearch
skip_if_unavailable=False

[fedora-debuginfo]
[fedora-source]
... analogous for debug and source code packages
```

The address of the package source can be specified either absolutely using `baseurl=...` or via `mirrorlist=...` in the form of a mirror file. This file contains a list of mirror servers. DNF chooses one of the mirrors independently. DNF replaces the variables `$releasever`, `$arch`, `$basearch`, and so on in the configuration file with the version number of the Linux distribution, its architecture, and other details. Let me briefly explain the origin of the three most important variables:

- `$arch` provides the architecture of the computer, such as `x86_64` for a 64-bit installation on Intel-compatible systems.
- `$basearch` is the basic architecture underlying `$arch` (also x86_64).
- `$releasever` results from the version number of the package whose name was specified in `dnf.conf` with the `distroverpkg` keyword. If this setting is missing, DNF uses the version number of the `fedora-release-common` or `redhat-release` package. For Fedora 38, `$releasever` simply contains the number 38.

Locking packages

If you want certain packages to be untouched by DNF and not updated when a new version is available, you need to add a line in `dnf.conf` or in the `*.repo` file of the package source; the line should contain `exclude name1 name2 name3`. You can use wildcards in the package names.

Plugins DNF can be extended with plugins and then can provide even more functions. The plugins are configured by files in the `/etc/dnf/plugins` directory.

16.3.3 Searching, Installing, and Updating Packages

Table 16.2 summarizes the most important actions you can perform with `dnf`. When you run the command for the first time, meta information about all the package sources set up is downloaded, which may take a while. All further commands are then executed immediately until the next update of the meta information is due.

Task	Command
Installing a package	`dnf install name`
Installing a local package file	`dnf localinstall file.rpm`
Determining the list of available updates	`dnf check-update`
Updating a package	`dnf update name`
Updating all packages	`dnf update`
Removing a package	`dnf remove name`
Obtaining a list of all repositories	`dnf repolist`
Obtaining a list of all installed packages	`dnf list installed`
Obtaining a list of all available packages whose name starts with `abc`	`dnf list available 'abc*'`
Searching for packages that contain the term `abc` in the package description	`dnf search 'abc'`
Editing package groups	`dnf grouplist/groupinstall/...`
Managing modules (AppStream)	`dnf module ...`
Displaying a list of last actions	`dnf history`
Determining details of action *n*	`dnf history info n`

Table 16.2 Important DNF Commands

Examples The following commands demonstrate the use of DNF on Fedora 38; the output has been shortened to save space:

```
root# dnf check-update
NetworkManager-config-connectivity-fedora.noarch   1:1.42.6-1.fc38
updates
amd-gpu-firmware.noarch                            20230404-149.fc38
```

```
updates
audit.x86_64                                    3.1.1-1.fc38
updates
...
root# dnf install mariadb-server
Installing:
 mariadb-server             x86_64   3:10.5.18-1.fc38  ...

Installing dependencies:
 mariadb                    x86_64   3:10.5.18-1.fc38
 mariadb-common             x86_64   3:10.5.18-1.fc38
 ...

Install  16 Packages, installed size: 117 M. Is this ok [y/N]: y
```

If you search for a package whose name you do not know exactly, it's best to run `dnf list available`. This command returns a list of all packages that can be installed. Using `grep`, you can filter the results:

```
root# dnf list available | grep mysql
```

Alternatively, `dnf search` can also be used. This command returns more hits because not only the package name but also the package description is taken into account:

```
root# dnf search mysql
```

Autocompletion

On Fedora, if you type `dnf install java` and then press Tab twice, a list of all package names beginning with `java` appears. This completion used to be lightning-fast, but unfortunately the process now takes several seconds.

Package groups

DNF has package groups that help you to install all the packages required for a specific task with little effort:

- A list of available package groups, including the group IDs (in parentheses), is provided by `dnf grouplist -v`.
- The `dnf groupinfo <grpid>` command tells you which packages belong to a group.
- `dnf groupinstall <grpid>` installs all mandatory and default packages of the group (but not the optional packages).

To update or remove a package group, you can use `dnf groupupdate` or `dnf groupremove`:

```
root# dnf grouplist -
v     (on RHEL 9, the group IDs in parentheses, shortened)
Installed Environment Groups:
   Server with GUI (graphical-server-environment)
Installed Groups:
   Container Management (container-management)
   Headless Management (headless-management)
Available Groups:
   RPM Development Tools (rpm-development-tools)
   .NET Development (dotnet)
   Console Internet Tools (console-internet)
   Graphical Administration Tools (graphical-admin-tools)
   Legacy UNIX Compatibility (legacy-unix)
   Scientific Support (scientific)
   ...
```

16.3.4 AppStream

An elementary problem of DNF used to be that only *one* version of a program could be provided in the package sources at a time. Developers in particular often need a different version, a more recent one with new features or an older one because their own code is only compatible with that specific version.

After several years of experimentation, Fedora has implemented the so-called modularity concept in version 28, which addresses precisely this weakness. In RHEL 8, this feature got a new name: *AppStream*.

Modules

AppStream and the associated DNF commands (see Table 16.3) are implemented around modules. Modules form groups of related packages.

Task	Command
Listing modules including versions	`dnf module list --all`
Listing installed modules	`dnf module list --installed`
Displaying module details	`dnf module info mname:n/profile`
Installing a module	`dnf module install mname:n/profile`
Installing a module (short notation)	`dnf install @mname:n/profile`
Uninstalling a module	`dnf module remove mname:n/profile`

Table 16.3 DNF Commands for Module Management

Task	Command
Selecting a module version (without installation)	`dnf module enable mname:n`
Canceling the version selection	`dnf module disable|reset mname`
Determining a module for a specified package	`dnf module provides pname`

Table 16.3 DNF Commands for Module Management (Cont.)

The difference compared to the package groups mentioned earlier is that modules can be available for selection in different version numbers as well as in multiple profiles. The module name is then followed by the version number and the profile in the following notation: `module-name:nn/profile-name`.

Examples

I tested the following examples on AlmaLinux 8 in spring 2023. You may ask: Why not on AlmaLinux 9 or another RHEL clone in the current version? Of course, RHEL 9 also supports streams, but at the time of testing, version 9 was still relatively new and there was not a single module that was available in two versions. `dnf module list --all` lists all modules available for selection. `dnf module list <name>` is clearer: This command returns only the modules with the same name. `[d]` marks the currently selected default variant:

```
root# dnf module list --all
Name                Stream          Profiles
389-ds               1.4
ant                 1.10 [d]        common [d]
container-tools      rhel8 [d]       common [d]
container-tools      1.0             common [d]
container-tools      2.0             common [d]
container-tools      3.0             common [d]
container-tools      4.0             common [d]
...
root# dnf module list php
php             7.2 [d]      common [d], devel, minimal
php             7.3          common [d], devel, minimal
php             7.4          common [d], devel, minimal
php             8.0          common [d], devel, minimal
...
```

To install PHP version 8.0 in the development profile, you want to run one of the following two commands. (The shorthand notation, in which the `@`

character identifies the name that follows as a module, is only permitted for the installation.) The selected module is *enabled* with the installation. If you later use `dnf install` to install additional packages from the module, the module that is now valid is automatically used without you having to explicitly specify it:

```
root# dnf module install php:8.0/devel
root# dnf install @php:12/server          (short notation)
```

If you later don't know which module a package came from, you can obtain this information using the `module provides` subcommand, where `[e]` stands for *enabled* and `[i]` for *installed*:

```
root# dnf module list --enabled php
php                 8.0 [e]         common [d], devel [i], minimal
```

`dnf module remove` removes all packages of a module. `dnf module reset` or (equivalently) `dnf disable` also resets the *enabled* state:

```
root# dnf module remove php:8.0/server
root# dnf module reset php
```

16.3.5 Additional Functions

Copr

Copr (*https://developer.fedoraproject.org/deployment/copr/copr-cli.html*) is an automated build system. Developers can use it to generate RPM packages from their own source code and then easily provide them to other Fedora or RHEL users as a package source. Copr thus has similarities with the better known PPA concept of Ubuntu.

At this point, I will only discuss the use of such Copr package sources. This is very simple. You set up the package source using `dnf copr enable` and then install the relevant package—here, for example, the *Remarkable* markdown editor:

```
root# dnf copr enable neteler/remarkable
root# dnf install remarkable
```

Source code packages

If you are interested in the source code of a package, you can download it using `dnf download --source`. This automatically activates the usually inactive `source` sources in the `*.repo` files. You can then install the resulting package `name.src.rpm` using `rpm -i`. Finally, you can find the source code in the `rpmbuild` directory in your home directory:

```
user$ dnf download --source nano
user$ rpm -i nano-n.n.src.rpm
```

Automatic updates

The `dnf-automatic` add-on package takes care of automatic package updates. The associated configuration file is `/etc/dnf/automatic.conf`. In it, you want to set `apply_updates` to `yes`. If you want, you can also set various email parameters to control who will automatically receive an email after each update:

```
# file /etc/dnf/automatic.conf
...
apply_updates = yes
```

A systemd timer takes care of the automatic updates:

```
root# systemctl enable --now dnf-automatic.timer
```

If you want to know when the next update is due, you should run `systemctl list-timers`:

```
root# systemctl list-timers '*dnf*'
```

Restart

Some updates only take effect when the computer is rebooted. Automatic reboots are not provided for in the Red Hat world. If you feel a desire for this behavior in your youthful recklessness, you can analyze the result of the `needs-restarting` command from the `dnf-utils` package in a Cron job. The command lists processes that should be restarted. Unfortunately, the kernel, whose update would usually be the most important reason for a reboot, is not covered:

```
root# needs-restarting | sort -n
1 : /usr/lib/systemd/systemd --system --deserialize 25
660 : /usr/sbin/postgrey --unix=/var/spool/postfix/postgrey/socket ..
965 : /usr/sbin/sshd -D
...
```

16.4 ZYpp

Like Fedora and Red Hat, SUSE uses RPM packages. However, the ZYpp package management, which is based on RPM, is a proprietary SUSE development. ZYpp stands for *ZENworks, YaST, Packages and Patches*, where the use of ZENworks is optional and only provided in SUSE Enterprise distributions.

libzypp

Behind the scenes, the `libzypp` library provides the basic ZYpp functions. `libzypp` can handle both YaST and Yum package sources. All configuration, database and cache files are located in the `/var/lib/zypp` directory. Both YaST and PackageKit rely on `libzypp` on openSUSE.

Updates versus Patches

`zypper` distinguishes between updates and patches! Updates are ordinary RPM packages that are available in a newer version than the installed one. Patches, on the other hand, are supplementary or update packages (delta RPMs).

SUSE uses patches as delta RPMs to update its own packages. External package sources, such as Packman, on the other hand, provide new package versions as updates.

Package sources

Package sources are stored in text files in the `/etc/zypp/repos.d` directory. If you modify these files in an editor, you must be sure to delete all backup copies afterward. Otherwise, you will get duplicates in the list of package sources. The following lines show the definition of the open-source package source for openSUSE:

```
# file /etc/zypp/repos.d/repo-oss.repo (openSUSE 15.5)
[repo-oss]
name=Main Repository
enabled=1
autorefresh=1
baseurl=http://download.opensuse.org/distribution/leap/$releasever/
repo/oss/
type=rpm-md
keeppackages=0
```

16.4.1 The zypper Command

`zypper` is a command interface to `libzypp`. `zypper` is thus the SUSE counterpart to `dnf` or `apt`. You can use it to search for, install, update, and remove packages, as well as manage package sources (see Table 16.4). `zypper` must be run by `root`.

Task	Command
Reimporting metadata of the package source	`zypper refresh`
Installing a package	`zypper install name`
Removing a package	`zypper remove name`
Obtaining a list of all updates	`zypper -t package list-updates`

Table 16.4 Important zypper Commands

Task	Command
Updating all packages	`zypper -t package update`
Performing a distribution update	`zypper dup`
Obtaining information on a package	`zypper info name`
Searching for packages whose package name contains `abc`	`zypper search abc`
Searching for packages whose description contains `abc`	`zypper search -d abc`
Deleting cached packages	`zypper clean`
Obtaining a list of all package sources	`zypper repos`
Setting up a new package source	`zypper addrepo uri name`

Table 16.4 Important zypper Commands (Cont.)

Examples

The following examples show how you can use `zypper`. The first command lists the package sources, the second one updates the sources, and the third one installs the `emacs` editor:

```
root# zypper repos --show-enabled-only
 #  Alias                 Name                      Enabled  GPG Check
Refresh
 3  repo-backports-update  Update repository of ...  Yes      Yes        Yes
 8  repo-non-oss           Non-OSS Repository        Yes      Yes        Yes
 9  repo-openh264          Open H.264 Codec          Yes      Yes        Yes
10  repo-oss               Main repository           Yes      Yes        Yes
12  repo-sle-update        Update repository ...     Yes      Yes        Yes
14  repo-update            Main Update Repository    Yes      Yes        Yes
15  repo-update-non-oss    Update Repository (Non ...  Yes      Yes        Yes
...
root# zypper refresh
All repositories have been refreshed.

root# zypper install emacs
The following 10 NEW packages are going to be installed:
  emacs emacs-info emacs-x11 etags libXaw3d8 libm17n0 libotf0 m17n-db
  m17n-db-lang system-user-games
10 new packages to install. Overall download size: 26.9 MiB.

Continue? [y/n] (y): y
```

Package groups To install all required packages for a certain task, such as to use the computer as a file server, ZYpp uses patterns. `zypper search -t pattern` retrieves a list of all such package groups. `zypper info -t pattern name` shows which packages belong to a package group. Using `zypper install -t pattern name`, you can install all packages of a package group.

History The `/var/log/zypp/history` file contains a very useful reference about when which package was installed or removed from which package source and what configuration work was done.

Distribution updates You can use `zypper dup` to perform a distribution update on the fly. However, you will need to take care of the necessary changes to the package sources yourself:

```
root# zypper update     (update for the previous version)
root# ...               (change package sources to the new version)
root# zypper dup        (replace old packages with new ones)
root# reboot            (reboot)
```

16.5 Debian Package Management (dpkg)

Debian packages are managed on two levels. This section describes the `dpkg` command, which is responsible for installing and managing packages at the lower level. `dpkg` is similar to `rpm`. The command can install, update, and remove individual packages, testing if all package dependencies are fulfilled.

Similar to `rpm`, `dpkg` fails to resolve unfulfilled package dependencies itself or load packages from package sources on its own. APT (*Advanced Package Tool*; see Section 16.6) fulfills precisely these tasks. It is based on `dpkg` and provides similar features as the DNF and ZYpp systems just presented. For the actual package management, there are three commands to choose from: `apt`, `apt-get`, and `aptitude`. When it comes to interactive work, `apt` is the most convenient option. However, the differences among the three commands are small.

Although this and the following section refer to Debian packages, the information applies to all Linux distributions that use this package format. Besides Debian, these include, for example, the Ubuntu family, Raspberry Pi OS, and Linux Mint. If you switch from an RPM-based distribution to a distribution with Debian packages, you will find an excellent overview of `rpm` commands and equivalent `dpkg` and `apt` commands on this page: *https://help.ubuntu.com/community/SwitchingToUbuntu/FromLinux/RedHatEnterpriseLinuxAndFedora*.

Metadata

dpkg manages comprehensive meta information for all packages (a package description, a list of all files in the package, dependency data, etc.). This data is in *Debian control* (dctrl) format. The dctrl-tools package contains various commands to perform queries in the dctrl data. man grep-dctrl provides a detailed description of these commands and concrete usage examples.

16.5.1 The dpkg Command

Table 16.5 provides an overview of the most important dpkg options. In practice, you will mostly use dpkg to determine information about installed or available packages.

Task	Command
Installing or updating a package	dpkg --install file.deb
Configuring a package	dpkg --configure file.deb
Removing a package	dpkg --remove package name
Removing a package entirely (also changed files)	dpkg --purge package name
Determining all installed packages	dpkg --list
Searching for packages whose package description contains abc	dpkg --list abc
Determining a list of all files of the package	dpkg --listfiles package name
Determining a list of all configuration files of the package	cat /var/lib/dpkg/info/name.conffiles

Table 16.5 Important dpkg Commands

dpkg --list displays a status code consisting of two or three letters (see Table 16.6). The two most common status codes are ii for a correctly installed package and rc for a removed package where the configuration files are still available. To remove rc packages entirely, you need to run dpkg --purge name. More details about package status and troubleshooting are provided by man dpkg.

Code	Meaning
First Letter	***Desired State***
u	Unknown

Table 16.6 Letter Codes of dpkg-list Results

Code	Meaning
i	Install
r	Remove
p	Purge
h	Hold
Second Letter	***Actual State***
n	Not installed (not)
h	Partially installed (half-installed)
u	Unpacked, but not configured (unpacked)
f	Installed but not configured (failed config file)
i	Fully installed (installed)
c	Only the configuration files are installed (config file)
t	Waits for the trigger of another package (trigger)
Third Letter (Optional)	***Error Code***
h	Package should not be changed (hold)
r	New installation required (reinstall required)

Table 16.6 Letter Codes of dpkg-list Results (Cont.)

Examples The following examples illustrate how to use `dpkg` in standard situations:

```
root# dpkg --install test.deb
root# dpkg --search /etc/sensors3.conf
libsensors-config: /etc/sensors3.conf
root# dpkg --listfiles joe
/.
/etc
/etc/joe
/etc/joe/ftyperc
/etc/joe/jicerc.ru
/etc/joe/jmacsrc
/etc/joe/joerc
...
```

`dpkg --list` returns a list of all installed packages. I have shortened the output of the following example somewhat for better readability:

```
root# dpkg --list 'cups*'
    Name              Version    Architecture  Description
ii  cups              2.4.2-3    amd64         Common UNIX Printing ...
ii  cups-browsed      1.28.17-2  amd64         OpenPrinting CUPS Filters
un  cups-bsd          <none>     <none>        (no description available)
ii  cups-client       2.4.2-3    amd64         Common UNIX Printing ...
ii  cups-common       2.4.2-3    all           Common UNIX Printing ...
ii  cups-core-drivers 2.4.2-3    amd64         Common UNIX Printing ...
...
```

Reconfiguring a Package

During the installation of Debian packages, installation and configuration scripts are automatically executed. For a few programs, there are also interactive setup programs that help with the individual configuration of the package—for example, with the basic configuration of the Postfix email server. If you want to repeat the configuration later, you can run `dpkg-reconfigure package-name`.

get-selections and set-selections

To quickly retrieve a sorted list of all installed packages, you want to run `dpkg --get-selections`. The package list contains less detailed information than that of `dpkg --list` and is therefore easier to read. If you redirect the list to a text file and save it, you can install all packages later on another machine by using `dpkg --set-selections`:

```
root# dpkg --get-selections
accountsservice                                 install
acl                                             install
...
```

hold status

The *hold* status (i.e., the third letter in Table 16.6) means that a package is not to be updated during an update. The following two commands show how you can put a package into *hold* and how to remove it from *hold*:

```
root# echo "<package-name> hold"    | dpkg --set-selections
root# echo "<package-name> install" | dpkg --set-selections
```

16.6 APT

APT is for Debian packages what DNF or ZYpp is for RPM packages: a high-level package management system that downloads packages automatically from package sources and resolves package dependencies automatically. The combination of Debian packages and APT currently results in probably the

most mature package management system for Linux. Ubuntu and Debian, among others, use it as the standard system for package management.

Like DNF and ZYpp, APT requires special package sources that provide meta information about the contents of the packages and their dependencies in addition to the DEB packages.

apt versus apt-get versus aptitude

For the actual package management, there are three commands to choose from: `apt`, `apt-get`, and `aptitude`. The commands are very similar and even share the same syntax for many operations. `apt` and `aptitude` are optimized for interactive use, while `apt-get` is better suited for script programming and the automated execution of package management commands, respectively.

Current Debian and Ubuntu systems have `apt` and `apt-get` installed by default; `aptitude` must be installed later if required. Ubuntu recommends the `apt` command for interactive package management. Regardless, `apt-get` is still more popular, if only because a large number of tutorials on the internet use `apt-get`. But you won't go wrong with `apt` and `aptitude` either. It's also no problem to use one command and then the other.

The following three commands are equivalent. They download and install the `name` package and any packages that depend on it:

```
root# apt      install name
root# apt-get  install name
root# aptitude install name
```

16.6.1 Configuration

The configuration of APT is carried out by the two `apt.conf.d/*` and `sources.list` files in the `/etc/apt` directory. Additional package source definitions can be found in the `sources.list.d` directory.

The `apt.conf.d/*` file usually contains only a few basic settings, which you mostly leave as they are specified by your distribution. The `sources.list` file is more interesting as it contains the APT package sources line by line. The syntax of each line looks as follows:

```
package type uri distribution [component1] [component2] [component3] ...
```

The package type is `deb` for ordinary Debian packages or `deb-src` for source packages. The second column specifies the root directory of the package source. Besides HTTP and FTP directories, APT also supports ordinary directories, RSH or SSH servers, and CDs or DVDs. The third column indicates the distribution. In the following listing, this is Ubuntu 23.04 (codename Lunar Lobster). All other columns indicate the components of the distribution

that can be taken into account. The component names depend on the distribution and the package source! For example, Ubuntu distinguishes among *main*, *restricted*, *universe*, and *multiverse* packages, while Debian differentiates among *main*, *contrib*, *nonfree*, and similar components.

The package sources mentioned first are preferred. Thus, if a particular package is available for download from multiple sources, APT downloads it from the first source. The following listing illustrates this; for space reasons, each entry was spread across two lines:

```
# file /etc/apt/sources.list (Ubuntu)
deb http://de.archive.ubuntu.com/ubuntu/ lunar           main
restricted universe multiverse
deb http://de.archive.ubuntu.com/ubuntu/ lunar-updates   main
restricted universe multiverse
deb http://security.ubuntu.com/ubuntu    lunar-security  main
restricted universe multiverse
```

The easiest way to make changes to `sources.list` is to use a text editor. Alternatively, you can also use a graphical user interface, such as `software-properties-gtx`.

Installing an APT key

For most APT sources on the internet, the metafiles describing the package sources are signed by a key. Furthermore, the APT content directories contain checksums for all packages. This control mechanism can be used to ensure that no package has been subsequently modified. However, this check only works if APT knows the public part of the key and can thus determine the authenticity of the packet archive.

All package keys are stored in the `/etc/apt/trusted.gpg.d` directory. If you want to use an additional package source, download the key file from the project page and copy it to this directory—for example, as follows:

```
user$ curl https://myrepo.com/myrepo.asc | \
      sudo tee /etc/apt/trusted.gpg.d/myrepo.asc
```

The associated package source must explicitly reference the filename of the key file by using `signed-by`:

```
# /etc/apt/sources.d/myrepo.list
deb [signed-by=/etc/apt/trusted.gpg.d/myrepo.asc] https://apt.myrepo.com ...
```

In the past, the `apt-key` command was used to manage APT keys, but this is now considered obsolete and should no longer be used. You can find more details on that at *https://askubuntu.com/a/1307181*.

16.6.2 apt Command

The actual package management can be carried out using the `apt` command or its alternatives, `apt-get` or `aptitude`. Ubuntu explicitly recommends `apt` for interactive use. I haven't found any information about this for Debian, but even there it's true that `apt` is more user-friendly and contains commands that are missing in `apt-get` and for which you would otherwise have to call `apt-cache` or `dpkg`. The most important `apt` commands are listed in Table 16.7.

Task	Command
Updating metadata from package sources	`apt update`
Installing a package	`apt install name`
Updating all packages	`apt upgrade`
As above, but uninstalling packages if necessary	`apt full-upgrade`
Removing a package	`apt remove name`
Uninstalling packages that are no longer needed	`apt autoremove`
Deleting cached packages from cache	`apt autoclean`
Listing all available packages	`apt list`
Displaying all installed packages	`apt list --installed`
Searching for a package	`apt search keyword`
Displaying package information	`apt show package-name`

Table 16.7 Important apt Commands

Examples

Before installing packages, you should run `apt update` to download the latest information from the package sources. This neither installs nor updates packages; it only concerns the package descriptions—that is, the metadata! Most other package-management systems (DNF, ZYpp) update this metadata on their own when needed, but this is just not true for `apt`, `apt-get`, and `aptitude`. You can then download and install a new package using `apt install`:

```
root# apt update
root# apt install dovecot-imapd
...

The following additional packages will be installed:
  dovecot-core liblua5.4-0
```

```
Suggested packages:
  dovecot-gssapi dovecot-ldap dovecot-lmtpd dovecot-lucene
  dovecot-managesieved dovecot-mysql dovecot-pgsql dovecot-pop3d
  dovecot-sieve dovecot-solr dovecot-sqlite dovecot-submissiond
  ntp ufw
The following NEW packages will be installed:
  dovecot-core dovecot-imapd liblua5.4-0
0 upgraded, 3 newly installed, 0 to remove and 32 not upgraded.

Need to get 6151 kB of archives.

After this operation, 14.4 MB of additional disk space will be used.

Do you want to continue? [Y/n] y
```

`apt remove package-name` removes the specified package. However, dependent packages originally installed together with the package remain unaffected. You can fix this via `apt autoremove`: this command removes all packages that are no longer needed.

Recommended packages

In addition to strict dependencies, some package descriptions also contain lists of other packages that are *suggested* or even *recommended*. The default behavior of `apt` is to ignore suggested packages, but to install recommended packages as well:

```
user$ apt-config dump | egrep -i 'recommend|suggest'
APT::Install-Recommends "1";
APT::Install-Suggests "0";
```

This behavior is not always appropriate. For example, I wanted to install the Emacs editor on Ubuntu 23.04. It turned out that the list of recommended packages includes the `postfix` mail server. This is absurd! If I need an editor, I don't want to add a mail server that might cause all kinds of security problems. The `--no-install-recommends` option provides a solution in such cases:

```
root# apt install --no-install-recommends emacs
```

Performing an update

The correct command to perform updates is usually `apt full-upgrade`. If required due to changed package dependencies, this will also install additional packages or remove existing packages. `apt upgrade` will also perform an update, but it will not involve packages if other packages need to be removed at the same time due to changed dependencies:

```
root# apt full-upgrade
```

Installing the source code

`apt source package-name` installs the source code of the relevant package into the current directory.

Distribution updates

To update to the next version of Debian or Ubuntu, you first need to adjust the package sources in `/etc/apt/sources.list` accordingly. In particular, you must replace the code name of the respective old version with that of the new version there; for example, when updating from Debian 11 to version 12, you must replace `bullseye` with `bookworm`. Then run `apt dist-upgrade`. The download and installation of the packages will take about half an hour, depending on the size of the installation. After that, reboot your computer, and you're done!

Unfortunately, release updates are a tricky business despite the basically excellent package-management system. The fact that all programs and server services really work as before after the update is more a case of luck than the rule. That's why I prefer a new installation most of the time.

16.6.3 The apt-get Command

The subcommands of the `apt-get command` are mostly the same as those of `apt`. Instead of `full-upgrade`, it's a bit misleadingly called `dist-upgrade`, although the version of the distribution does not change. The `list`, `search`, and `show` subcommands are not available. Instead, the `dpkg` and `apt-cache` commands provide comparable functions. `apt-get` is recommended primarily for the use in scripts or in Dockerfiles—that is, for noninteractive applications.

16.6.4 Additional APT Commands

tasksel

To install package groups, Debian and Ubuntu usually use meta packages: these are empty packages that just define a lot of package dependencies. For example, a whole collection of packages with basic development tools (compiler, `make`, etc.) is installed together with the `build-essential` meta package.

There is a second mechanism for defining package groups, which is based on the `tasksel` command. This mechanism is mainly intended to easily select package groups during the installation of the distribution, but `tasksel` can of course also be used later during operation. A list of all available package groups is provided by `tasksel --list-task`. To install package groups, you can use `tasksel install groupname`. If `tasksel` is executed without options, a dialog for selecting the desired package groups appears.

apt-cache

`apt-cache` retrieves various data about the available or already installed packages:

```
root# apt-cache show apache2
Package: apache2
Description: next generation, scalable, extendable web server
...
root# apt-cache  search scribus | sort
lprof - Hardware Color Profiler
scribus - Open Source Desktop Publishing
scribus-template - additional scribus templates
```

apturl

On Ubuntu, the `apturl` package is installed by default with the `apturl-gtk` program. This program allows you to install packages simply by clicking special `atp` links in the web browser after a query. You will come across such links mainly on Ubuntu wikis and forums.

History

All operations performed by the APT system (whether using `apt`, `apt-get`, or any other command) are logged in `/var/log/apt/history.log`. If this file does not exist or is empty, it may have been rotated (i.e., renamed and compressed). If necessary, you should then decompress `history.log.1.gz`, `history.log.2.gz`, and so on, and take a look at these files.

16.6.5 Automating Updates

unattended-upgrades

The `unattended-upgrades` package takes care of performing updates automatically, if they have been configured correctly. On Ubuntu, the package is installed by default; on Debian and Raspberry Pi OS, you need to perform the installation yourself.

While `unattended-upgrades` used to be started by a Cron job in the past, the `apt-daily-upgrade` systemd timer now takes care of the daily execution. You can determine the planned next execution time in the following way:

```
root# systemctl list-timers '*apt*'
```

The `unattended-upgrade` upgrade script then analyzes the `/etc/apt/apt.conf.d/*` configuration files.

Basic configuration

Unfortunately, the defaults in the configuration files are quite different depending on the distribution. Basic requirements are the following two parameters, which you may need to correct or add to a separate configuration file. I often use `99myown`. This file is processed at the end; its settings therefore take precedence over options from other files:

```
// file /etc/apt/apt.conf.d/99myown
// Update package list once a day
APT::Periodic::Update-Package-Lists "1";
// Actually perform updates
APT::Periodic::Unattended-Upgrade "1";
```

Which packages should be updated

Now you need to clarify which packages should be updated automatically. On Ubuntu, the `Allowed-Origins` parameter in the `50unattended-upgrades` file is preconfigured to install regular updates as well as security updates—a setting that makes a lot of sense:

```
// file /etc/apt/apt.conf.d/50unattended-upgrades (Ubuntu)
Unattended-Upgrade::Allowed-Origins {
    "${distro_id}:${distro_codename}";
    "${distro_id}:${distro_codename}-security";
    // also for Extended Security Maintenance, if available
    "${distro_id}ESMApps:${distro_codename}-apps-security";
    "${distro_id}ESM:${distro_codename}";
}
```

Debian and Raspberry Pi OS, on the other hand, control the packages to be updated through `Origins-Patterns`. The default configuration provides for the installation of security-critical updates only. If you also want to install normal updates, you must remove the `label` specification from the configuration file:

```
// file /etc/apt/apt.conf.d/50unattended-upgrades
// (Debian/Raspberry Pi OS)
Unattended-Upgrade::Origins-Pattern {
    "origin=Debian,\
     codename=${distro_codename},label=Debian";
    "origin=Debian,\
     codename=${distro_codename},label=Debian-Security";
    "origin=Debian,\
     codename=${distro_codename}-security,label=Debian-Security";
  };
```

To install all available updates from all package sources, you must modify the configuration as follows:

```
// file /etc/apt/apt.conf.d/50unattended-upgrades (Raspberry Pi OS)
Unattended-Upgrade::Origins-Pattern {
  "origin=*";
};
```

Using `Package-Blacklist`, you can exclude individual packages from the automatic update, if you want:

```
// do not update the following packages
Unattended-Upgrade::Package-Blacklist {
  "vim";
  ...
};
```

Kernel updates and reboot

In the course of the automatic updates, new kernel versions are of course also installed if required. However, such updates only take effect when the computer is rebooted. By default, this is not the case, but brave administrators can solve this with the following settings in `50unattended-upgrades` or in their own configuration files:

```
# if necessary, automatic reboot the next night at 2.00 am
Unattended-Upgrade::Automatic-Reboot "true";
Unattended-Upgrade::Automatic-Reboot-Time "02:00";
```

If you want to manually determine if a reboot is required, simply test for the existence of the `/var/run/reboot-required` file. `/var/run/reboot-required.pkgs` lists which packages require a reboot. Usually these are the kernel or system-related libraries. Alternatively, you can install the `needrestart` package and then call the command of the same name.

Logging

`unattended-upgrades` logs its efforts in the `/var/log/unattended-upgrades/unattended-upgrades` file. The file is especially of great help if `unattended-upgrades` does not work and you want to find out the reason. The timestamp files in the `/var/lib/apt/periodic` directory also show when certain operations were last carried out.

16.6.6 Synaptic

The most popular graphical user interface for the administration of Debian packages based on the APT commands was called Synaptic for a long time. Current versions of Debian and Ubuntu now provide the Software GNOME program for installing programs (as shown in Figure 4.14 in Chapter 4), but I personally still prefer the more technically oriented Synaptic program for finding specific packages (see Figure 16.2).

Installation

To install a specific package, double-click it to select it for installation. If the package depends on other packages, a dialog will appear with all other packages that also need to be installed.

The actual installation can be started by clicking the **Apply** button, where you still need to confirm a summary of all planned actions.

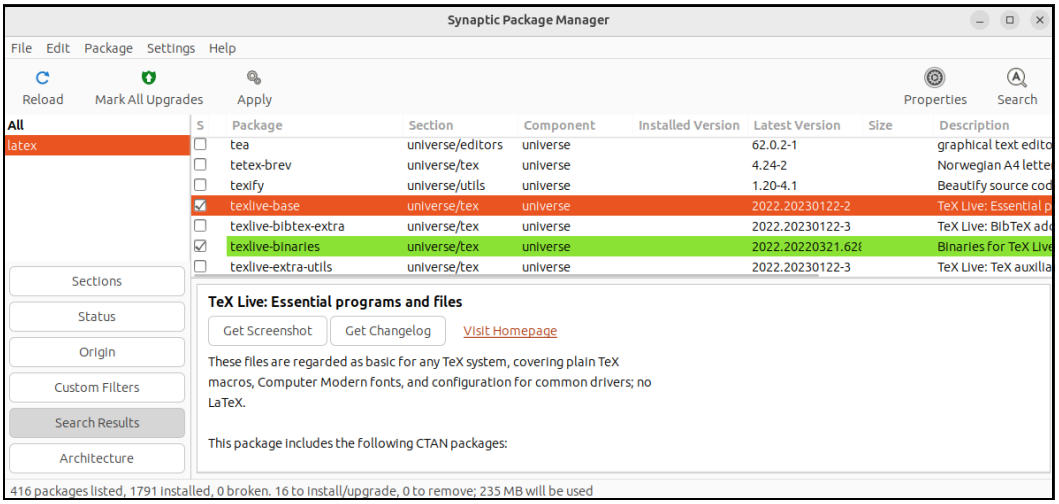


Figure 16.2 Package Management Using Synaptic

Defective packages

A package is considered "defective" if a problem occurs during installation or uninstallation and the process cannot be completed correctly. Synaptic and other package management tools refuse to work until this problem has been resolved.

To fix this, click the **Custom Filters** button in the page list in Synaptic and then on the **Defect** item. Synaptic then displays a list of all defective packages. Highlight all packages by pressing Ctrl+A, right-click the list, and select the **Mark for Reinstallation** entry. Then run the reinstallation by applying it. However, if you encounter problems again, mark the affected packages for removal.

Locking Problems

Only one package management program can run at a time. When trying to run two package management programs at the same time, the *unable to get exclusive lock* warning appears. This means that the program cannot access the internal package management files alone. To solve this problem, exit one of the two programs.

In rare cases, the lock warning occurs even when no other package management program appears to be running. The cause is usually that a program has not properly removed the `lock` file at the end of the program. If necessary, simply delete the `lock` file: `rm /var/lib/dpkg/lock`.

16.7 Pacman

Pacman stands for *Package Manager* and refers to the package management system for Arch Linux and derived distributions such as Manjaro. (Do not confuse Pacman with the openSUSE-specific package source, `packman`!) It is controlled by the `pacman` command, whereby, in contrast to the RPM and DEB world, there is no separation between a low-level and a high-level command.

Packages

Arch Linux packages are basically simple TAR archives that contain binaries of the respective program or library. The `*.pkg.tar.zst` identifier indicates the intended use as a pacman package and the compression via `zstd` (`tar` option `--zstd`). The package also contains some files with meta information about the package, such as `.BUILDINFO`, `.MTREE`, and `.PKGINFO`.

Package sources

Pacman and its package sources are configured through the `/etc/pacman.conf`, `/etc/pacman-mirror.conf`, and `/etc/pacman.d/*` files. By default, Pacman distinguishes among the following package sources:

- *Core*, for absolutely necessary basic packages
- *Extra*, for add-on programs, desktop environments, and so on
- *Community*, for packages provided by the community

In addition, there are optional package sources such as *multilib* for packages with 32-bit programs and libraries; *testing* for new, not yet sufficiently tested program versions; and *gnome-unstable* or *kde-unstable* for the development branches of KDE and GNOME. To enable an optional repository, it's sufficient to add two lines to `pacman.conf` according to the following pattern:

```
# Enable testing package source in /etc/pacman.conf file
...
[testing]
Include = /etc/pacman.d/mirrorlist
```

The name of the package source must be specified in square brackets. The following line can either refer directly to an HTTP server via `Server`, or to another file with this information via `Include`. `mirrorlist` contains a list of active Arch Linux mirror servers.

pacman

Package management is carried out using the `pacman` command. Unfortunately, the options of this command differ fundamentally from `apt`, `dnf`, and the like and are not easy to memorize (see Table 16.8).

Task	Command
Installing a package (*sync*)	`pacman -S --needed name`
Updating or reinstalling a package	`pacman -S name`
Installing a package from a specific repository	`pacman -S reponame/name`
Reimporting package sources	`pacman -Sy`
Updating all installed packages	`pacman -Su`
Updating package sources and packages (system update)	`pacman -Syu`
Searching packages (also not installed ones)	`pacman -Ss pattern`
Removing a package (*remove*)	`pacman -R name`
Removing a package plus all dependent packages	`pacman -Rs name`
Listing all installed packages (*query*)	`pacman -Q`
Displaying package information	`pacman -Qi name`
Listing files of the package	`pacman -Ql name`
Displaying the package that contains the file	`pacman -Qo filename`

Table 16.8 Important pacman Commands

The ubiquitous `-S` option stands for *sync*. In this respect, it's consistent that `pacman -S name` reinstalls the specified package in any case—even if the package has already been installed in the current version. To avoid unnecessary work, you should add the `--needed` option: then `pacman` only takes into account the packages that are not installed yet or installed only in an outdated version. Unfortunately, the option makes the frequently needed command unnecessarily bulky.

Even more options are documented in `man pacman`, as well as on the following web page: *https://wiki.archlinux.org/title/Pacman*.

Examples The following commands first perform a complete update of all package sources and installed packages and then install the Emacs editor including all required libraries. `--needed` prevents already installed packages from being installed again. `pacman -Qo` finds out which package provides the `/etc/resolv.conf` file. `pacman -Qi` then provides more information about this package:

```
root# pacman -Syu
root# pacman -S --needed emacs
root# pacman -Qo /etc/resolv.conf
/etc/resolv.conf is owned by filesystem 2023.01.31-1
root# pacman -Qi filesystem
  Name            : filesystem
  Description     : Base Arch Linux files
  Install Reason  : Installed as a dependency for another package
  ...
```

Arch User Repository

The *Arch User Repository* (AUR) is no ordinary Arch package source. Rather, it is a collection of build scripts on GitHub. Using AUR, you can compile software for which regular packages do not exist with relatively little effort. The basic requirement is to first install elementary development tools as well as `git`:

```
root# pacman -S --needed base-devel git
```

On the AUR website at *https://aur.archlinux.org*, you then search for the relevant package. For the following example, I use the `joe` package with the mini-Emacs clone, `jmacs`. Among other things, the description of this package contains the Git address. Using `git clone`, you now download the build file. `makepkg` downloads the rest of the source code, compiles the program, makes it a package file, and installs it (for the last step, you must specify your password for `sudo`):

```
user$ git clone https://aur.archlinux.org/joe.git
user$ cd joe
user$ less PKGBUILD     (short check)
user$ makepkg -si       (create and install local package file)
```

The disadvantage of AUR packages is that they cannot be updated using `pacman`. Rather, to perform an update, you need to change to the respective directory (`joe` in this case), use `git pull` to download the latest version of the code, and then repeat the build and installation process.

yay

There are so-called AUR helpers, commands that simplify the installation of AUR packages as well as their updates. One of these is `yay`. You can install it as follows:

```
user$ git clone https://aur.archlinux.org/yay-git.git
user$ cd yay-git
user$ mkpkg -si
```

From now on, you can install AUR packages simply by using `yay -S`:

```
user$ yay -S google-chrome
```

`yay` is a complete replacement for `pacman` and supports the same options, but also considers AUR packages for all actions. If you run `yay` without any options, the command automatically installs all available updates.

Marginal trust error

In rare cases, it happens that `pacman` or `yay` will refuse to work and will instead return a strange error message: *package signature from xxx is marginal trust*. This happens when the package keys, for whatever reason, have not been updated for too long and are out of date. `pacman` therefore cannot verify that the packages are authentic. The solution is to update the `archlinux-keyring` package and then update the package sources:

```
root# pacman -Sy archlinux-keyring
root# pacman-key --populate archlinux
root# pacman-key --refresh-keys
root# pacman -Syu
```

Cleaning up the cache

`pacman` and `yay` each manage a cache with downloaded packages and metadata. If you want to tidy up and create space, you should run one of the following commands:

```
root# pacman -Sc      (clear cache for uninstalled packages)
root# pacman -Scc     (clear cache also for installed packages)
root# yay -Sc         (also take yay cache into account)
```

Even more cleanup options are provided by the `paccache` command, which is well described in a blog post at *https://herbort.me/posts/automatically-cleaning-pacman-and-yay-cache-in-arch-linux*.

Graphic tools

You can also use the Pamac graphical user interface for package management. If you work on GNOME, this shell extension is an interesting alternative. The tool regularly checks whether updates are available and unobtrusively displays their number in the panel. Clicking there lists all affected packages and starts the update command in a terminal if necessary. For more information, see *https://extensions.gnome.org/extension/1010/archlinux-updates-indicator*.

16.8 PackageKit

PackageKit is a library for communication with various package management systems. The peculiarity of PackageKit is that it is compatible with several package management systems, including APT, DNF, Pacman, and

ZYpp. PackageKit thus creates a common, distribution-independent basis for user interfaces for package management and updates.

Gloomy Future

PackageKit has not been maintained since 2014. On the GNOME blog, developer Richard Hughes doesn't see much of a future for the project (*https://blogs.gnome.org/hughsie/2019/02/14/packagekit-is-dead-long-live-well-something-else*).

Nevertheless, PackageKit is still used by various programs in the desktop area—among others, by `gnome-software` and by Discover (KDE).

Internal details and configuration

PackageKit is configured using the `/etc/PackageKit/*` files. In the `PackageKit.conf` file, `DefaultBackend` can be used to set which package management system PackageKit should use, if required. However, PackageKit usually detects the backend automatically.

The `packagekitd` daemon is required to coordinate PackageKit operations. The program is automatically started by the PackageKit commands or user interfaces when required—and also terminated again when it is no longer needed. So the daemon does not run constantly as a background service.

Using the `pkcon` command, you can also perform all package operations in a console or automate them through a script. Note that the command must not be executed by `root`! You must start the command as an ordinary user. If necessary, the command uses PolicyKit to gain `root` privileges. If you want to track what PackageKit is doing in a console, you should run `pkmon`.

Cleaning up the cache

Over time, PackageKit stores countless files in `/var/cache/PackageKit` that are no longer needed later. The memory requirements for this directory can be several GiB. This can be solved by the following command, which updates the cache, but at the same time reduces the maximum age of the files to 0 (option `-c -1`):

```
root# pkcon refresh force -c -1
```

16.9 Firmware, BIOS, and EFI Updates

Usually, Linux package management takes care of keeping up to date all the software installed as packages. However, there is a special case that is outside the scope of this update mechanism: sometimes it happens that the BIOS or EFI of the motherboard has to be updated, or the firmware of the CPU, or a Wi-Fi adapter. There are usually security problems that are fixed in this way.

Intel was hit particularly hard recently. Almost all CPUs produced until 2019 have contained several massive security gaps at once (Spectre, Meltdown, etc.). When such hardware-related errors are fixed, a variable code area *within* a CPU (the microcode) is often updated. This is taken care of by the kernel, which transfers a suitable firmware file to the CPU immediately after startup. In addition, the kernel also includes custom code to work around the CPU bugs. However, this has nothing to do with the update mechanisms described here and is discussed in Chapter 21, Section 21.8.

In the past, most hardware manufacturers required Windows to update the firmware or BIOS/EFI, which is inherently unpleasant from a Linux perspective: While parallel installations of Windows and Linux are not uncommon in the private sector, pure Linux systems dominate in the server segment. There is no way to quickly start Windows to perform a necessary firmware update.

For this reason, various Linux and hardware companies have joined forces and founded the *Linux Vendor Firmware Service* (LVFS; *https://fwupd.org*). This initiative makes firmware updates available in a central repository in such a way that the updates can be used by Linux distributions without complications.

16.9.1 fwupd and fwupdmgr

The implementation of the firmware update service is based on two programs:

- The `fwupd` daemon runs in the background, determines the hardware components active in the system, and checks *https://fwupd.org* for updates.
- The `fwupdmgr` command helps to administrate and perform the updates. `fwupdmgr get-updates` downloads current metadata from the LVFS website. The `get-devices` and `get-updates` subcommands retrieve a list of all components covered by the update service and updates available for them. `fwupdmgr update` initiates the update, although the actual execution usually requires a reboot.

  Although `fwupdmgr` often recognizes the need for updates, there are no suitable update files available. One possible reason is that the hardware manufacturer does not work well enough with LVFS. However, there are also updates that `fwupdmgr` cannot perform, so you can only do them manually using vendor-specific tools.

The following lines show the application of the module (due to space constraints, the outputs are heavily shortened):

```
root# fwupdmgr refresh
root# fwupdmgr get-devices
Thunderbolt host controller:
  Device ID:          02dcd173032b1c4f136f185c0564b059a2e3d305
  Current version:    50.00
  Device Flags:       * Internal device
                      * Updatable
  ...

UEFI Device Firmware:
  Device ID:          2292ae5236790b47884e37cf162dcf23bfcd1c60
  Summary:            UEFI ESRT device
  Current version:    65550
  ...

SSD 970 EVO Plus 2TB:
  Device ID:          c6a0cfba7c7d81e253fce571e1d1e9f6003ae1c7
  Summary:            NVM Express solid state drive
  Current version:    2B2QEXM7
  ...
root# fwupdmgr get-updates
root# fwupdmgr update
Devices with no available firmware updates:
 * PM981 NVMe Samsung 2048GB
 * SanDisk SDSSDA-2T00
Upgrade available for Thunderbolt host controller from 39.00 to 50.00
Thunderbolt host controller and all connected devices may not be
usable while updating. Continue with update? [Y|n]: y
```

Caution with EFI Updates

Under certain circumstances, EFI updates can cause the boot process to no longer function correctly the next time the computer is started. You must make sure that the EFI directory is in the `EFI` partition. If you use UEFI Secure Boot, you need `shim` so that `fwupd` can pass the updates to the Secure Boot system. Before you start updating without a second thought, you should take a look at the following page: *https://wiki.archlinux.org/title/fwupd*.

GNOME Software and KDE Discover

You may have noticed that the Software GNOME package manager is not one of my favorite programs. However, the program does have at least one brilliant feature: it offers an uncomplicated user interface for the `fwupdmgr` command, making the installation of firmware updates a breeze, even on

Linux (see Figure 16.3). Comparable functions are also provided by the KDE package manager, *Discover*.

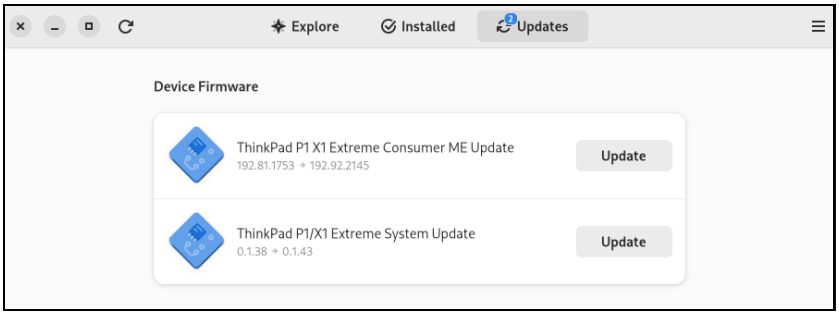


Figure 16.3 GNOME Software Manager Points to Various Updates for Lenovo Notebook

16.9.2 Internal Details of Microcode Updates

The microcode is transferred from the kernel to the CPU in the first few seconds. If the update is defective, the CPU does not run (properly) anymore and the boot process freezes. Usually, it isn't even clear what happened in the first place.

The microcode update is not stored permanently in the CPU but remains there only as long as the CPU is running. For this reason, the microcode must be transferred into the CPU again at every computer start. You can see whether microcode updates are performed during the boot process from the `dmesg` messages:

```
root# dmesg | grep -i microcode
  microcode: microcode updated early to revision 0xf0
  SRBDS: Mitigation: Microcode
  microcode: Microcode Update Driver: v2.2.
```

You can determine the active microcode revision number as follows:

```
root# cat /sys/devices/system/cpu/cpu0/microcode/version
0xec
```

The firmware files are located in the `/lib/firmware` directory on most distributions. The easiest way to find the packages responsible for microcode is to run `grep` through the list of installed packages (here for Debian/Ubuntu):

```
root# dpkg -l | grep microcode
ii  amd64-microcode  3.20...  amd64   Processor...AMD CPUs
ii  intel-microcode  3.20...  amd64   Processor...Intel CPUs
ii  iucode-tool      2.3....  amd64   Intel processor microcode tool
```

For Linux experts, there is even more background information on this topic in the following bug report: *https://bugs.launchpad.net/ubuntu/+source/linux/+bug/1829620*.

Microcode updates work by default on most distributions. If you use Arch Linux, you must install the `amd-ucode` or `intel-ucode` package, depending on your CPU architecture.

Firmware Glitches

Unfortunately, it has already happened that a microcode update for the CPU did not work. To prevent the kernel from patching the CPU during the boot process in the event of a microcode error, you can pass the `dis_ucode_ld` boot parameter to the kernel.

16.10 Managing Parallel Installations (Alternatives)

On Linux, there are often several alternative programs to choose from that serve the same purpose and sometimes even use the same command name: editors, Java environments, and so on. In some situations, it is convenient to install several variants or even several versions of one and the same program in parallel. As long as each program version ends up in its own directory tree, the installation itself is possible without conflicts. But which program version is used when the user executes a certain command?

To answer this question, many common distributions use a concept first developed by Debian, based on symbolic links in the `/etc/alternatives` directory. The following list indicates which package contains the `alternatives` directory and the associated `update-alternatives` management command:

- Debian, Ubuntu
  `dpkg` package
- Red Hat, Fedora
  `alternatives` package
- SUSE
  `update-alternatives` package

The easiest way to understand the concept is to take a look at an example. Let's assume that two Java versions are installed on one computer. Java programs are executed using `java class`. Now `/usr/bin/java` is implemented as a link to `/etc/alternatives/java`, while `/etc/alternatives/java` is another link pointing to the desired Java version:

```
user$ ls -l /usr/bin/java
...   /usr/bin/java -> /etc/alternatives/java
user$ ls -l /etc/alternatives/java
...   /etc/alternatives/java -> /usr/lib/jvm/java-17-openjdk-17.0.6/
bin/java
```

The links are usually managed automatically by scripts during the package installation. The `update-alternatives` command is used for this purpose. On Red Hat/Fedora, the command is also available under the `alternatives` name.

Listing alternatives

`update-alternatives --list` specifies for which programs/functions alternatives are maintained at all. This varies depending on the distribution.

You can use `update-alternatives --display` to determine which versions of a particular program are available and which version is by default. The following lines show the result for `editor` on a Debian system with multiple text editors installed. The `slave` lines concern commands that are subordinate to the actual program and `man` pages. `update-alternatives` automatically updates all `slave` links when the command link is changed:

```
root# update-alternatives --display editor
editor - auto mode
  link best version is /usr/bin/joe
  link currently points to /usr/bin/joe
  link editor is /usr/bin/editor
/bin/nano - priority 40
/usr/bin/jmacs - priority 50
/usr/bin/joe - priority 70
/usr/bin/jpico - priority 50
/usr/bin/vim.tiny - priority 15
...
```

Link management is usually carried out in automatic mode: Each installed package contains a priority number. `update-alternatives` activates the alternative with the highest priority for each (un)installation.

Selecting another alternative

`update-alternatives --config` determines the variant that will be active in the future. The command returns the list of alternatives available for selection, from which you then activate one. `update-alternatives` now updates the links. `update-alternatives --auto` returns to automatic mode if necessary. In the following example, `jmacs` is set as the default editor:

```
root# update-alternatives --config editor
There are 7 choices for the alternative editor (providing /usr/bin/
editor).
```

```
  Selection    Path               Priority   Status
* 0            /usr/bin/joe        70        auto mode
  1            /bin/nano           40        manual mode
  2            /usr/bin/jmacs      50        manual mode
  3            /usr/bin/joe        70        manual mode
  4            /usr/bin/jpico      50        manual mode
  5            /usr/bin/jstar      50        manual mode
  6            /usr/bin/rjoe       25        manual mode
  7            /usr/bin/vim.tiny   15        manual mode
Press <enter> to keep the current choice[*], or type selection
number: 2
using /usr/bin/jmacs to provide /usr/bin/editor (editor) in manual
mode
```

Internal management information about the links is stored in the `/var/lib/dpkg/alternatives` or `/var/lib/alternatives` directory, depending on the distribution.

16.11 Flatpak and Snap

I have already mentioned in the introduction to this chapter that package management systems based on DEB or RPM packages work well, but also have their limitations. For years, software vendors have been looking for a way to offer desktop applications across distributions in such a way that they can be easily installed by Linux users without extensive system knowledge. As so often in Linux history, it is becoming apparent that there is not one, but multiple solutions:

- Red Hat favors *Flatpak* (*http://flatpak.org*), which was developed in the GNOME environment and was initially called *xdg apps*. The Flatpak infrastructure is available by default on Fedora, Linux Mint, Pop!_OS, RHEL, and SUSE and can be installed on most other distributions.
- Canonical relies on its own *Snap* development (*https://snapcraft.io*) and ships the required packages with Ubuntu by default. Unlike Flatpaks, Snaps are also suitable for server use. Snaps are shared via the Snap Store, whereby all Snaps are currently free of charge.

Due to the high dominance of Ubuntu in the desktop segment, the Snap Store currently offers the larger and more popular range. Whether this is enough to prevail against Red Hat remains to be seen. In the past, many of Canonical's own developments have failed. This section covers Flatpak and Snap based on Fedora and Ubuntu, respectively.

Sandboxing

Even though there are many differences between Flatpak and Snap, there are also some common denominators. Both systems allow new applications to be installed at the user level, without the `root` privileges required for `apt/dnf/zypper`. At the same time, the programs installed in this way run in a "sandbox"—that is, with restricted rights. This is supposed to minimize security issues. Detailed descriptions of the Flatpak and Snap sandbox concepts can be found on the following pages:

- *https://github.com/flatpak/flatpak/wiki/Sandbox*
- *https://docs.ubuntu.com/core/en/guides/intro/security*

Space requirements

Flatpak and Snap have a fundamental disadvantage: the space required by programs installed with them is immense, often several times larger than with a conventional package installation. This is due to the fact that the cross-distribution execution of the programs can only be guaranteed if the programs bring along all necessary libraries in the correct version.

This is particularly extreme with Flatpak. There, many applications require a half-GNOME system on the side—even if your distribution uses GNOME anyway (possibly even in the right version)! If you are lucky, the libraries organized in their own packages are at least shared by several Flatpaks. However, with a bit of bad luck, different Flatpak packages use different libraries as their foundation.

Personal Assessment

With Flatpak and Snap, even Linux beginners can manage the "one-click installation" of popular programs such as Android Studio, IntelliJ, Slack, Spotify, VSCode, and so on. This is undoubtedly a huge step forward.

However, the new package formats are associated with an absurd waste of resources. A Spotify installation with Flatpak requires over 1.7 GiB of space on the SSD! The corresponding installation with Snap is almost economical at around 600 MiB. However, the comparison is flawed because various basic packages are preinstalled by default on Ubuntu and already inflate the distribution by around 700 MiB in the start state.

Note that the large space requirement does not only affect your disk. Because libraries cannot be shared with the rest of the system, running applications also requires more memory than traditional installations.

Sometimes there are also problems with functionality; that is, the interaction with programs installed in the conventional package system or (for security reasons) access to the file system or USB interfaces. In this regard, I had problems especially with development environments.

After reading this chapter, you should have enough Linux know-how to install packages manually using `apt` or `dnf` or to set up a Private Package

Archive (PPA) on Ubuntu. If the software you want is available as a conventional package, you should prefer this variant.

16.11.1 Flatpak

Setting up Flathub-Repo

Some distributions have Flatpak preinstalled but no package source set up. `flatpak remotes` will then return no result.

The largest Flatpak repository is currently *Flathub*. To activate it, you must run the following command:

```
root# flatpak remote-add --if-not-exists \
        flathub https://flathub.org/repo/flathub.flatpakrepo
```

Usage

Provided the Flatpak infrastructure is available, you can start the installation of a Flatpak-packaged application in the web browser by clicking a link that points to a `.flatpakref` file. The installation then takes place in the Software GNOME program.

flatpak command

Alternatively, you can install Flatpaks in the terminal by using the command of the same name. The following lines show how you can install and run Spotify:

```
root# flatpak install flathub com.spotify.Client
Required runtime for com.spotify.Client/x86_64/stable
(runtime/org.freedesktop.Platform/x86_64/22.08) found in remote
flathub
Do you want to install it? [Y/n]: y
..
org.freedesktop.Platform.GL.default  22.08         142 MB
org.freedesktop.Platform.GL.default  22.08-extra   142 MB
org.freedesktop.Platform.Locale      22.08         333 MB
org.freedesktop.Platform.openh264    2.2.0         944 kB
org.freedesktop.Platform             22.08         214 MB
com.spotify.Client                   stable        167 MB
```

For many applications, multiple Flatpaks are installed, including, in addition to the actual application, those for the required libraries. After all, these libraries are sometimes shared between multiple applications.

The `flatpak` command also helps with the subsequent administration of all installed Flatpaks: `flatpak list` provides a list of installations, `flatpak update` updates all Flatpaks as far as there are updates in the source of the respective package, and so on.

Using `flatpak uninstall`, you can remove Flatpaks again. User-specific data and configuration settings in `.var/app` are preserved unless you also pass the `--delete-data` option.

Internal affairs

Flatpak installations end up in the `/var/lib/flatpak` directory. The countless directories and files created there often use UUIDs as names—that is, long hexadecimal codes. Nearly 39,000 directories, files, and links were created for the Spotify installation, with a footprint of 1.7 GiB. It wasn't long ago that this was enough for an entire Linux distribution!

The execution of Flatpak programs requires that the `flatpak-session-helper` background process is running. This process, which is started by systemd with the rights of the respective user when logging in, communicates with the Flatpak commands and programs via the D-Bus system.

16.11.2 Snap

Usage

On Ubuntu, you can install Snaps directly in the Ubuntu Software package manager. This is basically very user-friendly. However, some programs now appear twice, such as the GIMP image-editing program. Only a close look reveals whether the package is a Snap package (**source = Snap Store**) or a conventional package (**source = ubuntu-xxx**).

By default, the package source is set to *https://snapcraft.io/store*. Because some Snap packages are preinstalled (most recently, for example, `core22` and `gnome-42` and the Firefox browser), installing another package for the first time does not involve such huge downloads.

snap command

In the terminal, you can administrate Snap using the command of the same name. `snap install` installs a new package, `snap list` lists all installed Snap applications, `snap run` starts an application, `snap refresh` updates all Snap packages, and `snap remove name` deletes a package:

```
user$ sudo snap install spotify
spotify 1.2.9.743.g85d9593d installed
user$ snap list
Name                    Version             Publisher    Notes
bare                    1.0                 canonical    base
core20                  20230404            canonical    base
core22                  20230404            canonical    base
firefox                 112.0.2-1           mozilla      -
gnome-3-38-2004         0+git.6f39565       canonical    -
gnome-42-2204           0+git.587e965       canonical    -
gtk-common-themes       0.1-81-g442e511     canonical    -
snap-store              41.3-71-g709398e    canonical    -
snapd                   2.59.1              canonical    snapd
```

```
snapd-desktop-integration  0.9                  canonical    -
spotify                    1.2.9.743.g85d9593d  spotify      -
```

Internal details

The internal management of Snaps is quite different from that for Flatpaks. Snap applications are stored as compressed `*.snap` files in `/var/lib/snapd/snaps`:

```
user$ ls -lh /var/lib/snapd/snaps
  ...    64M  Mai  5 09:38  core20_1879.snap
  ...    73M  Apr 18 22:36  core22_607.snap
  ...    73M  Mai  1 14:56  core22_617.snap
  ...   243M  Apr 21 14:58  firefox_2585.snap
  ...   244M  Apr 28 12:57  firefox_2611.snap
  ...   350M  May  5 09:39  gnome-3-38-2004_140.snap
  ...   461M  Apr 18 22:36  gnome-42-2204_87.snap
  ...   461M  May  1 19:41  gnome-42-2204_99.snap
  ...    92M  Apr 18 22:36  gtk-common-themes_1535.snap
  ...    54M  Apr 18 22:36  snapd_18933.snap
  ...   452K  Apr 18 22:36  snapd-desktop-integration_83.snap
  ...    13M  Apr 18 22:36  snap-store_959.snap
  ...   158M  May  5 09:39  spotify_64.snap
```

The Snap daemon `snapd`, which runs in the background, mounts these files as `squashfs` file systems at the `/snap/xxx` location in the directory tree, making the applications accessible (all sizes in MiB):

```
user$ sudo du -mcs /snap/*
 203    /snap/core20
 475    /snap/core22
1278    /snap/firefox
 903    /snap/gnome-3-38-2004
2289    /snap/gnome-42-2204
 360    /snap/gtk-common-themes
 192    /snap/snapd
  42    /snap/snap-store
 338    /snap/spotify
...
6079    total
```

Compared to Flatpak, the compressed flat images save space on the disk, but noticeably slow down the first start of a Snap app.

Updates

On Ubuntu, Snap packages are automatically updated like ordinary packages. The functioning of Snap can be influenced by some options, which are documented at *https://docs.snapcraft.io/system-options*.

Better safe than sorry?

By default, Snap saves a backup of each installed package with the previous version. Over time, this doubles the storage space used by Snap!

To delete packages that are no longer needed, you can write the following miniscript. `export LANG=` resets the language settings so that the output of `snap` is in English. The script removes all snap packages whose status is `disabled`:

```
#!/bin/bash
# file ~/bin/delete-snap-crap
# idea: https://superuser.com/questions/1310825
export LANG=
snap list --all | awk '/disabled/{print $1, $3}' |
    while read snapname revision; do
        snap remove "$snapname" --revision="$revision"
    done
```

You must run the following script with `root` privileges:

```
user$ sudo delete-snap-crap
```

For automation purposes, you should store the script in `/etc/cron.daily`.

Criticism of Canonical's Snap Store

While Flatpaks can basically be installed from arbitrary repositories, Snap packages are linked to the Snap Store run by Canonical. The server software used there is not available as open-source code. Canonical's control of the Snap Store is naturally a thorn in the side of other distributions and is another reason why there are currently few Snap supporters outside the Ubuntu world. For more information, see *https://blog.linuxmint.com/?p=3906*.

16.11.3 AppImages

Because Canonical relies on its Snap concept and Red Hat sees the future in Flatpaks, the chances that a third package concept will prevail are currently quite low. That's a shame because I think that AppImage (*https://appimage.org*) would be a charming alternative. An AppImage is a compressed executable file that internally contains the program along with all dependent libraries.

Benefits and drawbacks

Compared to Snaps and Flatpaks, no special requirements are needed on either the client or server side:

- AppImages can be offered for download like ordinary files; this makes app stores and the like superfluous.

- AppImages do not have to be installed explicitly. It is enough to save the downloaded file somewhere—even if it is in the `Downloads` directory. The `/home/name/Applications` or `/home/name/bin` directories are better suited.
- AppImages can be started by double-clicking. The desktop system only has to support the AppImage format, which is the case on both GNOME and KDE.

I'm generally a big fan of simple solutions, but admittedly the concept has its drawbacks: the security concepts developed for Snaps and Flatpaks (sandboxing) are missing, as are reasonable update mechanisms. (To update, you need to download the new version of the AppImage and delete the old one.) For beginners, it is inconvenient that AppImages are not included in the start menu of the respective desktop system.

AppImages hub

Basically, any website can offer AppImages for download. However, this sometimes makes it difficult to find AppImages. That is why there are some pages with links to programs that are offered as AppImages. Unfortunately, not all entries on these pages are up to date (see *https://appimage.github.io/apps*).

Usage

On most desktop systems, an AppImage can be started simply by double-clicking it in the file manager. If this doesn't work, you can set the execute bit in the terminal and run the program there:

```
user$ cd Downloads
user$ chmod +x myappname.AppImage
user$ ./myappname.AppImage
```

FUSE problem

AppImage mounts the image of the program into the local directory tree by using FUSE. (FUSE stands for *Filesystem in Userspace*; see Chapter 18, Section 18.7.) This allows the program to be used without first unpacking all the files.

Basically, FUSE works like magic. However, AppImage requires FUSE version 2, while some distributions have already upgraded to version 3. This applies, for example, to Ubuntu as of version 22.04. The best solution is to install FUSE 2 in parallel. On Ubuntu, this works as follows:

```
root# sudo apt install libfuse2
```

This allows the AppImage file to be started again simply by double-clicking it in the file manager. The alternative is to unpack the image file once. As a result, you need to run the `AppRun` program in the new `squashfs-root` directory:

```
user$ ./myappname.AppImage --appimage-extract
user$ ./squashfs-root/AppRun
```

You can find more information about the FUSE problem and distribution-specific tips at *https://github.com/AppImage/AppImageKit/wiki/FUSE*.

Internal details

If you want to inspect the contents of an AppImage (perhaps because you're curious or have security concerns), you should run the AppImage with the `--appimage-mount` option:

```
user$ ./myappname.AppImage --appimage-mount
/tmp/.mount_myappname_xxx
...
<Ctrl>+<C>
```

As long as the AppImage is running, you can take a look at the specified mount directory in a second terminal window by using `tree` or `lstree`:

```
user$ tree /tmp/.mount_myappname_xxx
...
440 directories, 2195 files
```

16.12 Distribution-Specific Characteristics

In this chapter, it is difficult to separate the general description of package management tools from the specifics of individual distributions. In the previous sections, I tried to explain the basic principles and commands that are common to several distributions. This section now describes some distribution-specific characteristics.

16.12.1 Debian

Package groups

Debian packages are divided into four groups:

- **Main**
  These are the basic Debian packages. The source code of these packages is available under a license that complies with the strict rules of the Debian project. This guarantees that their use and distribution is really free in the sense of the open-source idea.
- **Contrib**
  Packages of this group are also freely available, including source code. However, the packages can only be used in combination with *nonfree* packages. This applies, for example, to all programs that are based on libraries whose license is subject to restrictions in some way.

- **Nonfree**
  Packages of this group are free of charge, but their license is not in line with the Debian project's open-source ideal. For many nonfree packages, no public source code is available at all.
- **Nonfree firmware**
  This subgroup of nonfree packages is new in Debian 12. It only includes firmware files that are required for the operation of Wi-Fi adapters, graphics cards, and other hardware components. These files used to be in the initially nonactivated nonfree group, which caused a lot of problems—often as early as installation, when the drivers to connect to the wireless network were missing.

  Even though Debian still rejects nonfree drivers, it has pragmatically decided to provide the corresponding packages by default starting with Debian 12.

In addition, Debian distinguishes among *stable*, *testing*, and *unstable* packages:

- Only the packages that are part of the current, official Debian distribution are considered stable. These packages are usually stable and secure, but not particularly up to date.
- You can install more recent versions if you set up the unstable package sources. As the name implies, you are to some extent putting the stability of your system at risk. (But Ubuntu also relies mostly on unstable packages, so too much fear is not appropriate.) The total of the unstable packages represents the current Debian development state. No official updates are planned for the unstable branch. Known bugs are fixed simply by releasing a new version.
- The testing packages are intended as a transitional stage between stable and unstable, as it were. Unstable packages with no critical bugs detected for ten days automatically end up in testing (but only if all dependent packages are also free of critical bugs!).

Depending on the stage of development, there may also be temporary *experimental* packages to try out fundamentally new concepts.

Code names

All Debian versions have code names:

- Debian 10, Buster (2019)
- Debian 11, Bullseye (2021)
- Debian 12, Bookworm (2023)
- Debian 13, Trixie (approx. 2025)
- Debian 14, Forky (approx. 2027)

sources.list

To ensure that the APT system has access to all packages including updates, `sources.list` should look like the following example (I have spread the two statements across two lines each to save space):

```
# file /etc/apt/sources.list (Debian 12)
deb http://deb.debian.org/debian/ bookworm  main non-free-firmware
non-free contrib

deb http://security.debian.org/debian-security bookworm-security
main non-free-firmware non-free contrib
```

Note that `contrib` and `non-free` are usually not included after a fresh Debian installation. So if you want to have access to *all* packages, you must add these keywords yourself! If you also want to install source packages, you can copy these statements and replace `deb` with `deb-src` in each case.

CD/DVD package source

DVD drives are no longer fashionable. However, when installing in a virtual machine, the downloaded ISO image is used as a virtual DVD drive. After the installation, there is often a line in `sources.list` that starts with `deb cdrom`. Note that you must delete this line! Otherwise, Debian will attempt to access the DVD or its image that is no longer available after the installation is complete each time a package is installed.

16.12.2 Fedora

Additional package sources

To install proprietary drivers, multimedia codecs, and the like, you must set up the RPM Fusion package source (see Chapter 3, Section 3.2).

Distribution updates during operation

You can upgrade Fedora from one version to the next using the `dnf-plugin-system-upgrade` DNF module. For this purpose, you want to run the following commands:

```
root# dnf install dnf-plugin-system-upgrade
root# dnf update
root# dnf repolist --releasever 39
root# dnf config-manager --set-disabled repo-name
root# dnf system-upgrade download --releasever 39
root# dnf system-upgrade reboot
```

Instead of `39`, of course, you specify the appropriate version. You can use the `dnf repolist` command to determine whether all the package sources you set up in the previous Fedora version are also available for the current Fedora version. If not, you need to disable the package sources (`enabled=0` in `/etc/yum.repos.d/*.repo`). If you encounter problems during the update, you can try to temporarily uninstall the affected packages. Afterward, you can clean up using `dnf system-upgrade clean` and then start a new attempt

by using `dnf system-upgrade reboot`. You can find more details on that process at *https://fedoraproject.org/wiki/DNF_system_upgrade*.

Note that distribution updates are prone to errors. I have often had problems with them. If in doubt, you should perform a new installation!

16.12.3 openSUSE

Package management using YaST

On openSUSE, you can use YaST for package management. This configuration program provides five modules to choose from in the **Software** group. Especially SUSE newcomers sometimes find it difficult to find the right one among the similar entries. The following overview provides initial guidance, and the most important modules are presented in more detail ahead:

- The obsolete **Media Verification** module tests whether an installation CD or DVD is free of errors.
- **Online Update** launches YOU (*YaST Online Update*) to download updated packages or security updates. If you work on KDE or GNOME, updates can alternatively be performed with desktop-specific programs based on PackageKit.
- **Install and Delete Software** leads to the central package management module, which allows you to install new SUSE packages, remove or update existing ones, and so on. This is probably the module you will use most often.
- **Software Repositories** helps to manage the package sources. Via **Add • Community Repositories**, you can set up popular package sources with a few mouse clicks. The two most important package sources are **Packman** (current multimedia packages) and **NVidia Graphics Drivers**.
- **Add-on Products** enables the installation of mostly commercial programs that are offered on a disk or in package sources on the internet. This module is *not* intended to install common packages belonging to SUSE; for that, you must use **Install Software**!

Installing software

There are two fundamentally different implementations for the YaST **Software • Install and Delete Software** module. The KDE variant corresponds to what SUSE users have been used to for many years. The newer GNOME variant offers the same functions but is a bit different to use. In the following, I refer to the KDE version (see Figure 16.4).

In the main window, you can use the **Show** button to open different views (dialog sheets) and switch between them:

- **Search**
  In this view, you can search for packages whose name or function you know.

- **Installation Summary**
  In this view, you can see which packages are currently marked for installation, update, or removal.
- **Package Groups**
  This view displays packages that match in content—for example, all games.
- **Patterns**
  This view shows packages that have a functional similarity—for example, all packages for setting up a web server (see Figure 16.4). This allows you to quickly and conveniently select all packages that perform a specific task for installation. Basically, **Patterns** and **Package Groups** follow the same idea, the difference being that the grouping logic is different.
- **Languages**
  This view shows all localization packages for a specific language.

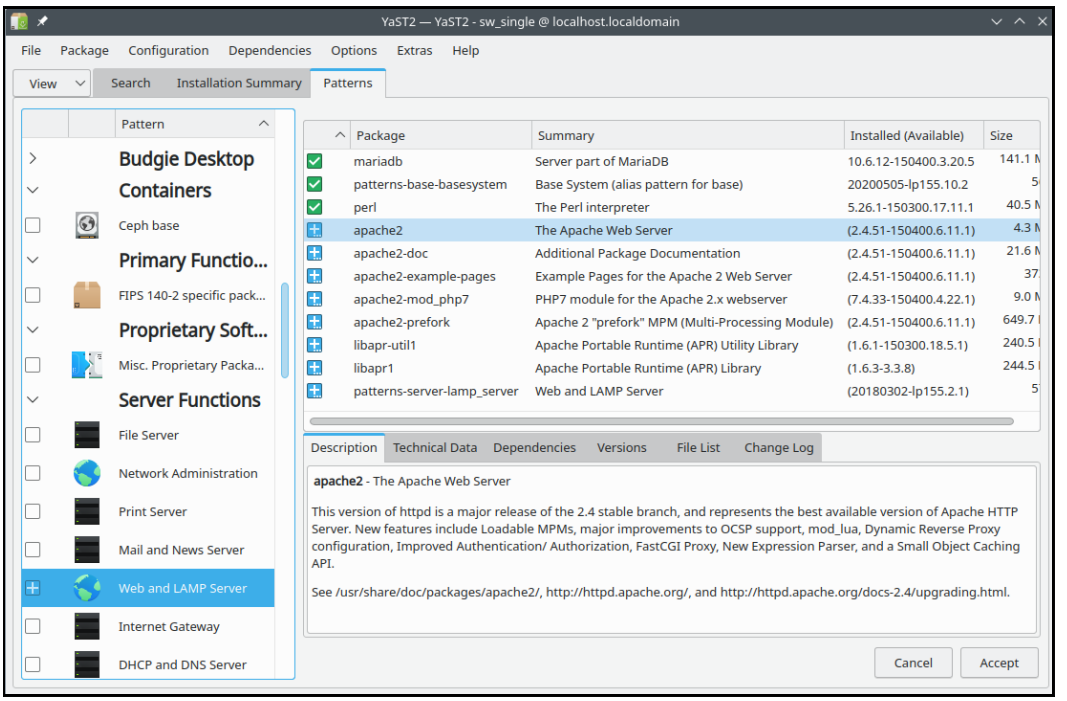


Figure 16.4 Installing Software Packages via YaST on KDE

The package manager checks the package dependencies during each installation and activates additional packages for automatic installation or update if necessary. If dependency conflicts occur, YaST displays various suggestions for how to fix the problem.

The status of packages is expressed by symbols. The state can be changed using the mouse (use the context menu with the right mouse button if necessary!) or the keyboard. You can get a complete description of all symbols via **Help • Symbols**.

Automatic updates

In addition to the five existing YaST software modules, you can install a sixth one:

```
root# zypper install yast2-online-update-configuration
```

After a YaST restart, you can use the **Online Update Configuration** module to configure automatic updates on a regular basis.

Distribution updates

openSUSE provides two ways to perform a distribution update:

- **Using an installation medium**
  This update variant is similar to a new installation. You reboot the computer from a USB flash drive. The installation program detects the existing openSUSE version and provides its update as an option. However, the system is *offline* during the update, which takes about half an hour.
- **During running operation**
  To perform a hot update, you must first delete the existing repositories and replace them with new repositories for the current openSUSE version. Then run `zypper dup` in a console. After that, a restart is required. Version-specific tips for dealing with `zypper dup` can be found on the following website: *https://en.opensuse.org/SDB:Zypper_usage#Distribution_upgrade*.

One-click install (*.ymp files)

The *one-click install* method allows for one click in the web browser on a YMP file to permanently set up the listed package sources and install all specified packages. Behind the scenes, a YaST module takes care of this work. Before starting the installation, you must of course enter the `root` password. The YMP files are simple XML files that contain all the necessary information (package sources, package names, and so on). YMP stands for *Yast Meta Package*.

You can also install one-click links as `root` using the `OneClickInstallCLI` command (formerly OCICLI):

```
root# OneClickInstallCLI "https://example.com/super-cool-app.ymp"
```

16.12.4 RHEL and Clones

subscription-manager

On RHEL, all package management features are locked until you have unlocked your installation using `subscription-manager` (see Chapter 22, Section 22.2). So you won't get updates or package installations without a

license. This restriction only applies to RHEL, not to clones like AlmaLinux or Oracle Linux.

CRB package source

Since RHEL 8, there is a new package source called *CodeReady Linux Builder* (CRB) with a wide collection of development tools for various programming languages, from Java to Go and Rust. However, this package source is not active by default. To change this, you can enter `enabled=1` in the corresponding repository file or, even more simply, run the following command:

```
root# dnf config-manager --set-enabled crb
```

EPEL package source

Even after enabling the CRB repository, the package selection in RHEL is modest compared to other distributions. The situation looks much better right away if you set up the EPEL (*Extra Packages for Enterprise Linux*) package source. It contains a huge collection of well-maintained add-on packages. On AlmaLinux or Rocky Linux, you can set up EPEL effortlessly with the following command:

```
root# dnf install epel-release          (AlmaLinux, Rocky Linux)
```

On RHEL and Oracle Linux, you must download the relevant package:

```
root# dnf install https://dl.fedoraproject.org/pub/epel/ \
  epel-release-latest-9.noarch.rpm   (Oracle Linux, RHEL)
```

EPEL 10

There are plans to restructure EPEL for RHEL 10 and offer separate branches for each minor version: for 10.0, 10.1, and so on. However, it's currently unclear when RHEL 10 will be introduced (presumably in 2024 or 2025) and whether the EPEL plans will actually be put into practice by then. The new concept and its motivation are described here: *https://discussion.fedoraproject.org/t/epel-10-proposal/44304*.

REMI package source

Another recommended repository is REMI (*https://blog.remirepo.net*), which provides up-to-date PHP packages. Because RHEL now offers multiple PHP versions itself thanks to AppStream, the importance of REMI has decreased somewhat. However, if you want to use a new PHP version quickly after its release, REMI is still the first choice. It can take up to six months before the PHP version is released as an AppStream.

Update philosophy

Currently, there is no Linux distribution that promises updates over a longer period than RHEL. With the updates, you upgrade your distribution step

by step from version 9.0 to 9.1, 9.2, 9.3, and so on. However, an update to the next major version, such as to 10.0, is not planned.

In this method, mainly security updates are carried out; only in exceptional cases will there be version jumps. For selected package groups, however, different versions are offered as modules (AppStreams).

Interestingly, the kernel version also remains unchanged over the lifetime of the distribution. On this point, however, you should not be deceived by the version number: Red Hat developers are constantly building important security updates and, on a case-by-case basis, new features into the kernel. In this way, Red Hat tries to combine maximum stability with some modernity.

Note that some RHEL clones ship their own kernel versions. This is especially true for Oracle Linux, which offers its Unbreakable Enterprise Kernel in a significantly more up-to-date version than Red Hat.

Repo analysis

`dnf repolist` lists all active repositories. If you want to know which installed packages are from a particular repository, you should combine `dnf list installed` with `grep`:

```
user$ dnf list installed | grep @epel
epel-release.noarch     9-5.el9          @epel
exim.x86_64             4.96-5.el9       @epel
fail2ban.noarch         1.0.2-3.el9      @epel
joe.x86_64              4.6-12.el9       @epel
...
```

`dnf repoquery -i` tells you which repositories provide a particular package, in which version, and for which CPU architecture:

```
user$ dnf repoquery -i zlib
Name         : zlib
Version      : 1.2.11
Release      : 34.el9
Architecture : i686
Repository   : baseos
...
```

To count the number of packages a repository provides, you want to run `dnf list available`, using options to first disable all repositories and then reenable the repository in question:

```
user$ dnf list available --disablerepo="*" --enablerepo="epel" | wc -l
14985
```

16.12.5 Ubuntu

Package sources

There are four Ubuntu package groups, all of which are enabled by default in `/etc/apt/sources.list`:

- **Unrestricted support (main)**
  These packages are part of Ubuntu, are freely available and can be freely redistributed without licensing issues. `main` packages are maintained and provided with updates by the Ubuntu team.
- **Restricted copyright (restricted)**
  `restricted` packages contain programs that are important for the functioning of Ubuntu Linux but are not available as open-source software. These are in particular hardware drivers for graphics and Wi-Fi cards. Also, the `restricted` packages are officially supported and maintained by Ubuntu. For security updates, however, the Ubuntu team relies on the support of the companies that provide the respective programs.
- **Community maintained (universe)**
  `universe` packages contain open-source programs that are not maintained by the Ubuntu team. Instead, members of the Ubuntu community take care of these packages.
- **Nonfree (multiverse)**
  `multiverse` packages contain programs or data that are not under an open-source license or that do not comply with Debian's rules for free distribution. Like `universe` packages, these packages are not maintained by Ubuntu.

Personal Package Archive

A PPA is a way for Ubuntu developers to make current versions of various programs available without officially integrating them into the Ubuntu package sources. PPAs often provide the fastest way to integrate new versions of graphics drivers, LibreOffice, GIMP, and the like into Ubuntu in a relatively safe way. You can find more information about PPAs at *https://launchpad.net/ubuntu/+ppas*.

To set up a PPA package source and install a package from it, you can simply run the following commands:

```
user$ sudo add-apt-repository ppa:name
user$ sudo add-get update
user$ sudo add-get install package-name
```

Support status

The most appealing feature of the Ubuntu LTS distributions is their five-year update guarantee. However, it should be noted that the long maintenance period only applies to packages from the main and restricted groups! In real life, however, it's often necessary to install packages from the universe or multiverse groups or from completely different package sources.

This makes it increasingly difficult to keep track of which packages still have updates.

The `pro security-status` command from the `ubuntu-advantage-tools` package provides initial help in this regard. When called without any other parameters, it provides an overview of how many packages are being serviced and for how long:

```
root# pro security-status
1776 packages installed:
1732 packages from Ubuntu Main/Restricted repository
  36 packages from Ubuntu Universe/Multiverse repository
   8 packages from third parties

This machine is receiving security patching for Ubuntu Main/Restricted
repository until 2027.

This machine is NOT attached to an Ubuntu Pro subscription.
```

Ubuntu Pro

Ubuntu Pro is a paid add-on for commercial customers for LTS versions that extends the support period to 10 years and also includes the universe package source. Private customers can use Ubuntu Pro free of charge for up to five installations.

If you want to try Ubuntu Pro, you need an account with Ubuntu One. This will allow you to log in to the Ubuntu Pro Dashboard (*https://ubuntu.com/pro/dashboard*). There you will receive a free token that can be used for five installations. You can use `pro attach` to activate Ubuntu Pro on your computer:

```
root# pro attach 4alkj23r8...
```

Behind the scenes, `pro attach` sets up two new package sources:

```
user$ ls /etc/apt/sources.list.d/ubuntu-esm*
/etc/apt/sources.list.d/ubuntu-esm-apps.list
/etc/apt/sources.list.d/ubuntu-esm-infra.list
```

Free Kernel Updates with One Important Limitation

`pro attach` also activates the kernel live patch feature (see Chapter 21, Section 21.6). Note, however, that Canonical is using all free users of Ubuntu Pro as beta testers! They receive kernel updates earlier than paying customers. If no problems occur, paying customers will receive the same updates a little later.

> Basically, there are few objections to this approach. I have never had any problems with such beta updates. However, it's annoying that Canonical doesn't transparently point out the beta character of the free kernel updates.

Proprietary drivers

The Applications and Updates program (`software-properties`) supports you in installing proprietary drivers. After starting the program, you want to switch to the **Additional Drivers** dialog sheet. The program then analyzes your hardware and, if necessary, provides suitable drivers for installation (as shown in Figure 17.2 in Chapter 17).

In an emergency, such as if the graphics system fails, you can also perform the driver installation in text mode using the `ubuntu-drivers` command:

- `ubuntu-drivers devices` shows which drivers are available for which hardware components.
- `ubuntu-drivers list` displays the currently active drivers.
- `ubuntu-drivers autoinstall` performs an automatic installation or update of all suitable drivers.
- `ubuntu-drivers debug` displays the status of the drivers and error messages, if any.

However, the command does not provide a way to manually select drivers. To install or remove a driver in a different way from `ubuntu-drivers autoinstall`, you must install the relevant packages yourself using `apt`:

```
root# ubuntu-drivers devices
== /sys/devices/pci0000:00/0000:00:01.0/0000:01:00.0 ==
modalias : pci:v000010DEd00001CBBsv000017AAsd00002267bc03sc00i00
vendor   : NVIDIA Corporation
model    : GP107GLM [Quadro P1000 Mobile]
driver   : nvidia-driver-525 - distro non-free recommended
driver   : nvidia-driver-515 - distro non-free
driver   : nvidia-driver-510 - distro non-free
driver   : nvidia-driver-515-server - distro non-free
driver   : nvidia-driver-470 - distro non-free
driver   : nvidia-driver-470-server - distro non-free
driver   : nvidia-driver-390 - distro non-free
driver   : nvidia-driver-450-server - distro non-free
driver   : nvidia-driver-525-server - distro non-free
driver   : xserver-xorg-video-nouveau - distro free builtin
root# apt install nvidia-driver-460
```

Language packages

The Languages program (`gnome-language-selector` command from the `language-selector-gnome` package) allows you to install or complete language packages for the current language or for another language and set the default language. The change to the default language will take effect at the next login. However, the program only takes care of programs that directly belong to Ubuntu. If you also have KDE programs installed, you need to install the `kde-l10n-en` package for their localization.

Distribution updates

Usually, the update system only updates individual programs, not the entire distribution. As soon as the update system detects a new Ubuntu version, it asks whether it should perform a full distribution update. Do not answer **Yes** thoughtlessly! Distribution updates take a relatively long time and are often associated with problems.

If necessary, you can also start the distribution update manually using `do-release-update -m desktop` (for desktop systems) or `-m server` (for servers). For Ubuntu LTS versions, release updates are only intended for the next LTS version—for example, from 22.04 to 24.04. If you want to upgrade to a non-LTS version, you must first set the `Prompt` variable from `lts` to `normal` in `/etc/update-manager/release-upgrades`.

Chapter 17
Graphics System

The Linux graphics system is currently experiencing an upheaval. After several decades of using the *X Window System* (not *windows*, with a plural *s*!) as a basis, the transition to the successor system Wayland started in 2017. Fedora has been using Wayland by default for several years, RHEL since version 8, Debian since version 10, and Ubuntu since version 21.04.

Wayland is therefore the standard for modern installations. However, the X window system remains installed in parallel and serves as a fallback in case of driver problems. In the application, the change from X to Wayland should be invisible: everything looks the same as before, and ideally everything should work the same as before. In the longer term, the changeover to Wayland promises performance benefits and easier programming. Although it is clear that the importance of X will diminish, this chapter is still very much focused on X. On the one hand, X is still used if the graphics driver is not Wayland-compatible. This especially affects the proprietary NVIDIA driver. When I updated this chapter in May 2023, the NVIDIA drivers were basically Wayland-compatible—and actually worked reasonably well, even though final stability problems still remain. On the other hand, thanks to the XWayland internal compatibility layer, conventional X programs continue to run on Wayland.

17.1 Basic Principles

If you have worked with Windows or macOS so far, the differentiation between graphics and desktop systems is difficult to understand. Both terms seem to refer to the same thing. A Windows PC or a Mac *always* runs in graphics mode and displays a desktop system that contains windows.

Linux is different in this respect: in the server or IoT area, it is quite common that there is no graphics mode at all and operation occurs only via the network or via text commands.

But there are also big differences in desktop installations. The graphics system running in the background, be it Wayland or X, is combined with a desktop system. On a Fedora installation, Wayland is used together with GNOME. And on openSUSE you have Wayland or X in combination with KDE. If Linux systems look very different, this has primarily to do with the desktop system. You must therefore differentiate between the low-level

graphics system on the one hand (Linux kernel including drivers for the graphics card, Mesa 3D library, and X and/or Wayland) and the high-level applications based on it on the other (GNOME, KDE, and all programs that run in graphics mode). X or Wayland only do very basic tasks like communicating with the drivers or accepting mouse, trackpad, or keyboard input. The actual appearance and behavior of a Linux desktop is not influenced by this; it is the libraries and programs based on X or Wayland that are responsible.

17.1.1 The X Window System

The X Window System (X for short) was originally developed by the Massachusetts Institute of Technology. X describes basic functions for drawing points, rectangles, and so on, but also a network protocol that makes it possible to execute an X program on computer A and display the results on computer B.

X server

The X server establishes the interface between the X window system and the hardware. The server is modularized; this means that the actual server is supplemented by a module with the specific functions for the respective graphics card. The X server is also responsible for processing input events, such as pressing a key or moving the mouse.

X extension modules

The standard functions of the X server can be extended by various additional modules (extensions), which are responsible, for example, for 3D graphics, for video output, and so on.

Window manager

The window manager is an X program responsible for managing the windows. You can use the window manager to start other programs, switch between windows, move and close windows, and so on—in other words, to perform tasks that are actually quite trivial.

The window manager is also responsible for the decoration of the windows—that is, for the design of the frame around the actual window content, including the title bar with buttons for closing or maximizing the window. It is important to keep in mind that these tasks are carried out by the window manager and not by X itself. KDE and GNOME each have their own window manager.

Compositor

Modern desktop systems equip windows with a discreet shadow, perform animations when opening and closing windows, enable transparency effects, and so on. In KDE and GNOME, the compositor functions are integrated in the window manager. However, it is technically possible to separate the window manager and the compositor and implement them separately. This hasn't always been the case.

17.1.2 Wayland

Wayland is not a program, but "only" a protocol for the communication between the *Wayland compositor* (a display server) and the application programs (clients). The GTK library used by the GNOME project is Wayland-compatible from version 3, the Qt project for KDE from version 5. For programs that do not yet use Wayland libraries themselves, there is the intermediate *Xwayland* layer, which establishes the compatibility to the X Window System.

Wayland partially uses the same kernel-based mechanisms or drivers as X. These include the processing of inputs (`libinput`), the setting of the resolution (*Kernel Mode Setting* [KMS]), and access to the video memory (*Direct Rendering Manager* [DRM]). In this respect, Wayland does not reinvent the wheel, but draws on many modules that are also used on X. In contrast to X, communication with the 3D functions of the Mesa library takes place via the EGL interface.

The break with X and the abandonment of full compatibility to countless old specifications has the advantage that a lot of code from X that is no longer relevant for practical use does not have to be maintained any further. In this respect, Wayland promises to be a much leaner system.

Wayland compositor

While on X the window manager is responsible for the decoration of the windows, in Wayland the compositor takes over this task. However, the compositor is not part of Wayland, but must be implemented by the respective desktop system—for GNOME by the GNOME shell, for KDE by Plasma and KWin. Wayland is much slimmer than X, so it provides fewer features. With the switch to Wayland, these functions have to be implemented by the respective desktop system.

Desktop systems

When GNOME runs on Wayland, Mother acts as a Wayland compositor. However, when GNOME runs on X, Mother serves as the window manager. In this respect, Mother is a bridge that is built between GNOME and X or between GNOME and Wayland, depending on the graphics system. Mother is not executed as a standalone program, but in the form of libraries to which the GNOME shell is linked.

On KDE, Plasma acts as the Wayland compositor.

Desktop systems like Xfce, which only have a small team of developers, are in danger of falling by the wayside in the long run. The implementation of a custom compositor and other Wayland conversions require a lot of work.

Software stack

Figure 17.1 summarizes the main components of the graphics system (image idea: Thorsten Leemhuis, c't 11/2021). You can see that the functions previously executed by the X server are replaced by the Wayland compositor as well as Wayland-compatible client libraries (Qt, GTK).

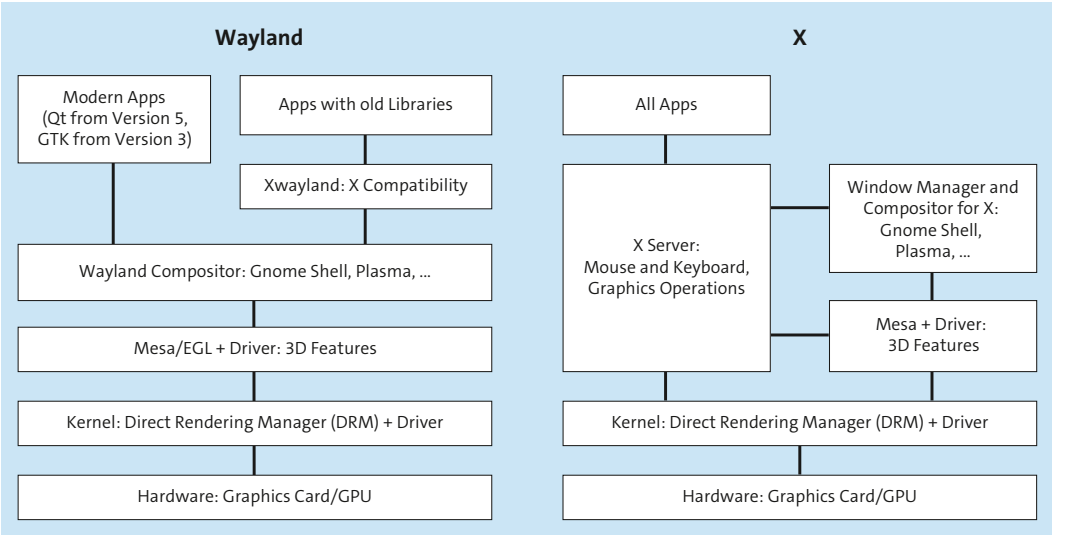


Figure 17.1 Diagram of Major Components of Wayland or X System

Wayland is the future

Hector Martin, lead developer of Linux drivers for computers running Apple's own CPUS (Apple Silicon), has written a blazing, highly technical appeal as to why there is no way around Wayland (see *https://social.treehouse.systems/@marcan/110371565062371963*).

17.1.3 Wayland Limitations Compared to X

Early Wayland adopters had to cope with massive limitations. Meanwhile, the situation has greatly improved, especially if you use a current distribution on GNOME. Remote maintenance or screen sharing is now basically possible as is the start of graphics programs with `root` privileges (e.g., `sudo gedit`). However, there are still some limitations:

- Theoretically, the proprietary NVIDIA driver is Wayland-compatible. However, the stability in interaction with Wayland was not satisfactory in my tests. Many distributions therefore automatically fall back to X if the proprietary NVIDIA driver is in play.

  As an alternative, you can use the `nouveau` driver. This driver is Wayland-compatible, but sometimes causes other problems—all the more so the newer the graphics card is.

- In principle, Wayland is not network-compatible. However, thanks to Xwayland, it's still possible to run an X-compatible program via SSH on host 1, but display and operate it via SSH on host 2. For modern programs, screen sharing and similar tasks can also be implemented using

the PipeWire library and Xdg-desktop-portal interface. This only works really smoothly in exceptional cases at the moment, but there is land on the horizon.

- Many programs that want to record or split the screen do not work. For example, the Linux version of Microsoft Teams causes problems. Initially, this was due to a general Wayland restriction. However, programs optimized for Wayland are quite capable of tapping the screen. For example, the TeamViewer program is Wayland-compatible with a few restrictions.
- Various X-specific tools such as `imwheel`, `xdotool`, `xkill`, `xrandr`, or `xmodmap` can either no longer be used or only work for programs executed via Xwayland.

17.1.4 Glossary

Texts about the Linux graphics system are teeming with abbreviations and obscure terms. This section provides guidance in alphabetical order.

DRI and DRM

The *Direct Rendering Interface* (DRI) allows X to use the 3D functions of the graphics card—provided there is a suitable DRI driver for the card. DRI builds on the *Direct Rendering Manager* (DRM) included in the kernel. Wayland also uses DRM, but via the EGL interface.

EGL

EGL is an interface between OpenGL and Wayland's window management. EGL allows Wayland programs to directly call 3D functions (OpenGL functions) to display the window contents. X programs use GLX instead of EGL.

GLX and libGL

On X, the OpenGL functions are used via the GLX library. This library establishes the connection between the X window system and OpenGL. For example, GLX ensures that Open GL output occurs only in the currently visible part of a window and does not collide with other windows. GLX is integrated into X through a module.

KMS

Kernel Mode Setting (KMS) means that the Linux kernel can set the graphics mode. KMS is supported by all major open-source drivers. KMS makes it possible to set the desired graphics resolution immediately after starting the computer and to start the desktop system a few seconds later without flickering.

Mesa and OpenGL

OpenGL (often called GL for short) is a library originally developed by SGI for displaying 3D graphics, which is available on almost all Unix/Linux computers. Almost all 3D programs and games available on Linux are based on OpenGL. In a way, OpenGL is the Unix/Linux counterpart to Microsoft's DirectX library.

As the OpenGL code was originally not freely available, the compatible open-source library Mesa was created. Mesa is thus responsible for the 3D functions. The library is positioned between the kernel and the graphics drivers on the one side and the Wayland or X programs on the other. Mesa was initially a pure software solution, but now uses the 3D functions of the graphics card.

OpenCL The *Open Computing Language* (OpenCL) is an interface for developing algorithms that can be parallelized and executed particularly efficiently by the GPU of a graphics card.

RandR The *Resize and Rotate Extension* (RandR) allows you to change some settings of the graphics system during operation. These include the resolution, frame rate, and image rotation. A second screen can also be activated via RandR.

VDPAU The *Video Decode and Presentation API for Unix* (VDPAU) is an API for decoding video streams. VDPAU was originally developed by NVIDIA but is also supported by AMD.

Vulkan Vulkan is an alternative to OpenGL that has been available since 2016. The 3D API is especially optimized for high speed. The design is similar to that of Direct3D 12 (Microsoft). Current Linux graphics drivers, Mesa and Wayland, support Vulkan.

XRender The *X Rendering Extension* (XRender) is a library for achieving transparency and overlay effects (*alpha blending*). The library is also used for text output. XRender relies on 3D hardware features for speed.

17.2 Graphics Drivers

Before I describe the configuration and operation of X and Wayland in the following sections, I want to discuss the biggest problem of both graphics systems here: the lack of support for modern NVIDIA graphics cards by open-source drivers.

The vast majority of all current PCs and notebooks contain graphics chips (*graphical processing units* [GPUs]) from the following three companies: AMD, Intel, and NVIDIA.

The good news is that there are open-source drivers for all three GPU types. The drivers for Intel and AMD work excellently, while the nouveau driver for NVIDIA GPUs is mediocre to good (the older your hardware the better). The speed achieved is not always ideal; sometimes 3D, additional, or energy-saving functions remain unused. The graphics driver can therefore be a

reason why a notebook computer runs significantly longer on Windows than on Linux.

For computers with NVIDIA graphics, there is therefore often no way around the proprietary NVIDIA driver. This driver is free, works well, and can now be set up easily on most distributions, often during the installation process. Nevertheless, this driver has various disadvantages, which I will discuss in more detail in the course of this section; the poor Wayland support is unfortunately only *one* issue.

Driver stack

Strictly speaking, there is not *one* driver, but multiple drivers. Together, they form the driver stack—that is, drivers for various components of the graphics system that build on each other:

- Kernel-level hardware drivers
- Direct rendering manager (`libdrm`)
- 3D functions/OpenGL/Volcano (Mesa)
- X driver (`xf86-video-xxx`)

The fourth level is only relevant when X is involved. In Wayland, this layer is omitted; apart from that, Wayland uses the same drivers as X.

Note that the names in the different layers are not consistent. The Intel driver is roughly composed of the `i915` kernel module, the `libdrm_intel` library, the `i965_dri` or `i915_dri` mesa driver (depending on the chip), and the `intel` X driver.

Benchmark tests and a lot of background information on the latest graphics drivers can be found on the following excellent website: *https://phoronix.com*.

Very detailed (although partly already somewhat older) information for a better understanding of the driver levels can be found on the following page: *https://blogs.igalia.com/itoral/2014/07/29/a-brief-introduction-to-the-linux-graphics-stack*.

17.2.1 Drivers for AMD, Intel, and NVIDIA

The following paragraphs summarize the current driver situation, sorted alphabetically by graphics card manufacturer.

AMD

Traditionally, there have been two drivers for graphics cards from AMD (formerly ATI) for many years: the `radeon` open-source driver and AMD's binary `fglrx` driver. Both drivers have been replaced by the new `amdgpu` open-source driver, which has been developed by AMD in collaboration with the open-source community since 2015 and is supplied with all common distributions.

For various reasons, AMD does not feel able to support all features of its GPUs through open-source drivers. These features include OpenCL, Vulkan, and VDPAU. Most Linux users will never miss these features. If you do, you need the `amdgpu-pro` driver. This extension to the `amdgpu` driver is available for free download from AMD's website. However, the driver is proprietary and the source code is not available. The following link refers to version 22.40 which was current in May 2023; by the time the book is published, there will certainly be newer versions: *https://amd.com/en/support/kb/release-notes/rn-amdgpu-unified-linux-22-40-3*.

Intel

Basically, Intel also works well with the open-source community. The Intel driver named `i915` is in the official kernel code; proprietary drivers do not exist. In this respect, buying a notebook or PC with an Intel CPU that includes an integrated graphics unit almost guarantees uncomplicated use on Linux. The performance is perfectly sufficient for ordinary desktop use.

The `i915` driver also supports the new, dedicated Intel graphics cards (brand name Arc). Its use requires at least the Linux kernel version 6.2.

Admittedly, the driver situation is not flawless at Intel either. The X-specific component of the Intel driver stack (`xf86-video-intel`) has not been maintained for some time. Many distributions therefore use the generic `xf86-video-modesetting` driver, which works better. If you use Wayland, this part of the driver is not relevant anyway.

NVIDIA

Unfortunately, NVIDIA still insists on its position that license agreements with other companies and patents would make the development of an open-source driver impossible and prevent public documentation of the internal interfaces. Instead, NVIDIA provides the free binary `nvidia` driver. Although its quality was good in the past, the basic disadvantages of a non-open-source driver remain (see Section 17.3). In addition, it took NVIDIA a very long time to make its driver Wayland-compatible. The situation got much better by 2023, but depending on the graphics card or GPU model, the combination of Wayland and `nvidia` drivers can still cause trouble (as of spring 2024).

Despite NVIDIA's resistance, the open-source community has developed its own driver, `nouveau`, which is now used by default in many distributions and works quite well for both X and Wayland. However, it is slower than NVIDIA's proprietary driver for various use cases and does not optimally utilize energy-saving features.

Your experience with the `nouveau` driver will be better the older your GPU model is. On my somewhat older Lenovo notebook with an NVIDIA Quadro P1000 mobile GPU, `noueveau` works so well that I can do without the proprietary driver by now.

Hybrid solutions

On many notebook computers and occasionally also on desktop computers, there are *two* graphics systems: an energy-saving system, which is usually integrated directly into the CPU, and a second system for high 3D performance. NVIDIA has named its implementation of this concept *Optimus*.

This hybrid approach attempts to combine a high runtime with high graphics performance—depending on what the user needs at the time. With the appropriate drivers on Windows, the active graphics system can be changed during operation without the user noticing.

This does not work in this way on Linux, but at least there are various solutions for changing the active GPU. Pop!_OS provides corresponding menu commands, but they require a reboot of the computer. Using the `prime-select` command (see Section 17.3), the GPU changeover succeeds without rebooting, but you have to log off and log on again. The best documentation on this subject can be found (as so often) on the Arch Linux wiki: *https://wiki.archlinux.org/title/NVIDIA_Optimus*.

Purchase recommendation

The basic rule is simple: if at all possible, avoid notebooks with NVIDIA graphics! Unless you want to mine Bitcoins, run games or password crackers, or do other GPU-intensive specialized tasks (AI research), the graphics performance of Intel or AMD CPUs with the integrated graphics features is absolutely sufficient.

17.2.2 Problems of Nonfree Drivers

You may be of the opinion that the distinction between "genuine" open-source drivers and free manufacturer drivers (also known as proprietary drivers, binary drivers, or restricted drivers) is splitting hairs; the main thing is that it works. However, there is good reason to prefer open-source drivers over binary drivers:

- Graphics drivers require a tight integration with the Linux kernel. For this purpose, there is a small kernel module between the actual driver (closed-source) and the kernel (GPL), which only serves as an interface. Many Linux developers have doubts that this approach is GPL-compliant and tolerate it only reluctantly. The kernel developers designate the kernel as *tainted* as soon as a non-GPL driver is loaded and refuse any support for problems in this case.
- After each kernel update, the kernel module of the graphics driver must also be updated. Ideally, the new graphics driver is automatically downloaded and installed by the package management system; if an error occurs during this process, in the worst case the graphics system no

longer works after the kernel update and you have to install or compile a new kernel module for the driver in a text console.

- If a security problem occurs in the driver, Linux distributors can only hope that the graphics companies will come up with an update as quickly as possible. With open-source code, on the other hand, the developer community can fix the bugs themselves, which is usually faster.
- Graphics support under Linux depends on the goodwill of the manufacturer. Older graphics cards are often not supported for very long.
- If you use UEFI Secure Boot and your distribution allows only signed kernel modules, you cannot use proprietary graphics drivers.

In the long run, Linux can only remain an open-source system if the most important components are also freely available, and this undoubtedly includes the graphics drivers. You should therefore also choose your next computer or graphics card with a view as to whether there are free drivers for it!

17.3 NVIDIA Driver Installation

Even though NVIDIA does not collaborate well with the Linux community, the company develops and builds outstanding graphics cards that are also popular in the Linux world. The `nouveau` open-source driver is only sufficient in basic cases. The optimal use of the graphics card usually requires the proprietary NVIDIA driver. Fortunately, its installation has become increasingly easier in recent years.

Installation Requirements

Before you start installing Linux on a notebook computer with an NVIDIA GPU, you must disable UEFI Secure Boot in EFI/BIOS. Secure Boot requires that only signed components be used throughout the boot process. This affects the boot loader, the kernel and all modules loaded by the kernel. As the kernel modules required for the NVIDIA driver are not available as source code, this condition cannot be met.

Ubuntu and Pop!_OS

Pop!_OS and Ubuntu provide a particularly high level of convenience for NVIDIA users. In both distributions, the NVIDIA drivers already migrate to your SSD during the installation phase. Pop!_OS provides the drivers directly on the installation media.

With Ubuntu, you must enable the **Install Third-Party Software** option in the setup program; the drivers will then be downloaded from the internet.

But even at a later stage, the driver installation is easy on Ubuntu as well as on distributions derived from it. To install the drivers, start the **Applications & Updates** program. The drivers suitable for your system are listed in the **Additional Drivers** dialog sheet (see Figure 17.2). To activate a driver, you need to restart the computer, and that's all!

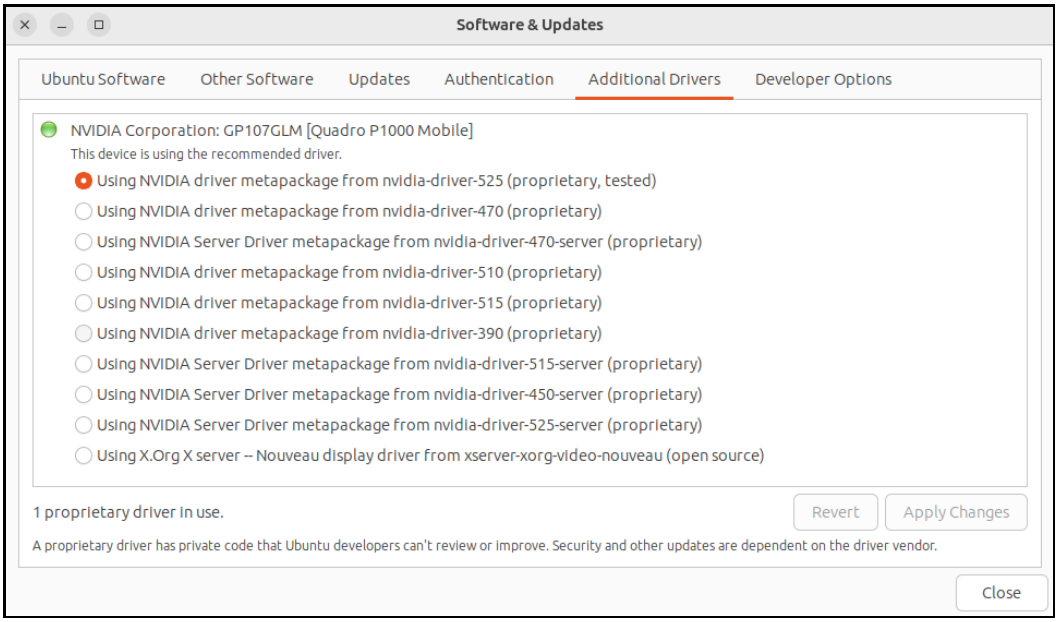


Figure 17.2 Installing NVIDIA Driver on Ubuntu

Fedora and RHEL

In Fedora, the **Software Sources** menu item in the Software program opens a dialog in which you can activate package sources for important proprietary programs. One of them is **RPM Fusion—Nonfree—NVIDIA Driver**. It contains the NVIDIA drivers. After activating this package source, you can install the NVIDIA driver and the associated configuration program directly in Software (see Figure 17.3). Basically, this procedure also applies to RHEL. However, in this case, you must manually set up the RPM Fusion package source (see *https://rpmfusion.org/Configuration*).

Popular alternatives to the RPM Fusion packages among professionals are the packages from the *negativo17.org* package source. There are various versions of NVIDIA drivers to choose from, as well as packages for additional functions such as CUDA (*Compute Unified Device Architecture*). A detailed description of the packages and their usage can be found at *https://negativo17.org/nvidia-driver*.

The first reboot takes a relatively long time because the NVIDIA kernel modules are compiled in the process. Press Esc if you want to read the init

messages. The NVIDIA driver becomes active without any further configuration work.

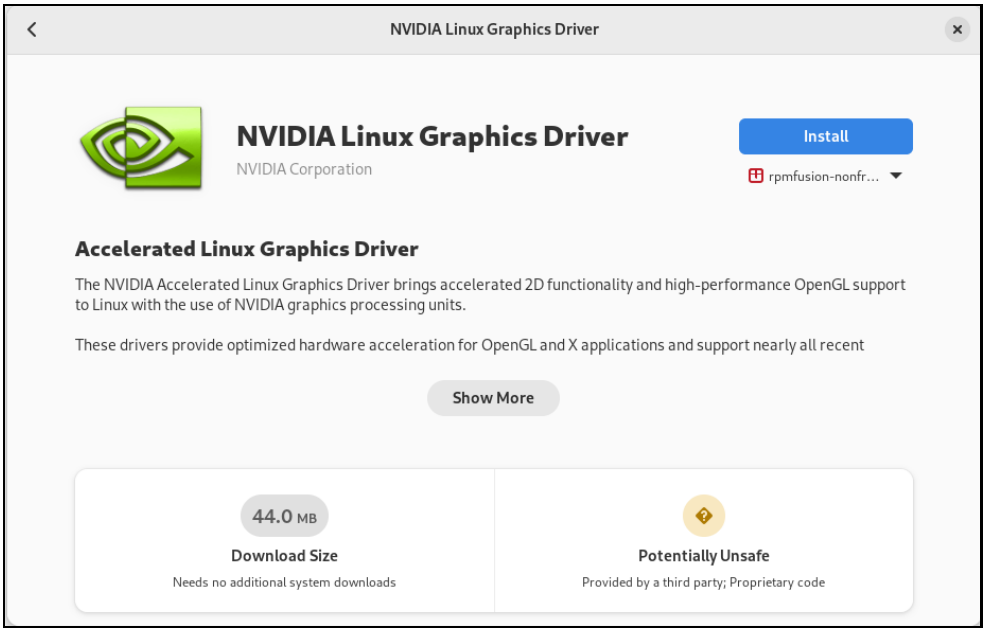


Figure 17.3 Installing NVIDIA Driver on Fedora

ArchLinux, openSUSE

ArchLinux provides the NVIDIA drivers as an AUR package. The installation is easiest with an AUR helper, such as `yay -S nvidia`.

On openSUSE, you can activate the NVIDIA repository and can then install the drivers using YaST. See the following pages for more info:

- *https://wiki.archlinux.org/title/NVIDIA*
- *https://en.opensuse.org/SDB:NVIDIA_drivers*

Manual installation

Only in an absolute emergency should you download the NVIDIA drivers directly from the NVIDIA site and install them manually. Even if the installation is successful, there is a considerable risk that this state will only last until the next kernel update.

17.3.1 Operation and Configuration

After installing the NVIDIA driver, you must reboot your computer. Usually, everything is now configured to load the driver automatically. You can check this yourself using `lspci -k | grep -i nvidia` or `lsmod | grep nvidia`, or by taking a look at the `Xorg.0.log` file:

```
user$ lsmod | grep nvidia
nnvidia_uvm           1024000  0
nvidia_drm              57344  4
nvidia_modeset        1228800  7 nvidia_drm
nvidia               34136064  365 nvidia_uvm,nvidia_modeset
drm_kms_helper         245760  2 nvidia_drm,i915
drm                    552960  10 drm_kms_helper,nvidia_drm,i915
user$ lspci -k | grep -i nvidia
01:00.0 VGA compatible controller: NVIDIA Corp. GP107GLM [Quadro
P1000 Mobile]
    Kernel driver in use: nvidia
    Kernel modules: nvidiafb, nouveau, nvidia_drm, nvidia
```

The `/lib/modprobe.d/nvidia*.conf` configuration files are responsible for the correct loading of the `nvidia` kernel modules and for the nonloading of the `nouveau` kernel modules. If Linux continues to load the `nouveau` driver, you should add the following lines to the `/etc/modprobe.d/blacklist.conf` file, then reboot the computer and try again:

```
# /etc/modprobe.d/blacklist.conf
blacklist nouveau
```

Only in exceptional cases do you need to modify `xorg.conf` to make it mandatory to use the `nvidia` driver:

```
# file /etc/X11/xorg.conf
Section "Device"
    Identifier  "Device0"
    Driver      "nvidia"
End
```

nvidia-settings

In simple cases, you can continue to use the settings program of your desktop to set up the graphics system (multimonitor setup, projector configuration, etc.). However, if you have special requests, you will have to launch the `nvidia-settings` program instead (see Figure 17.4). Some distributions require you to explicitly install the package of the same name beforehand.

`nvidia-settings` allows you to change numerous options immediately, without restarting. If the result is satisfactory, you can save the changes in `xorg.conf`. You must then specify the `root` password or your own password for `sudo`. Also note that some settings—especially those for monitor configuration—are only available when you use X, but not in combination with Wayland.

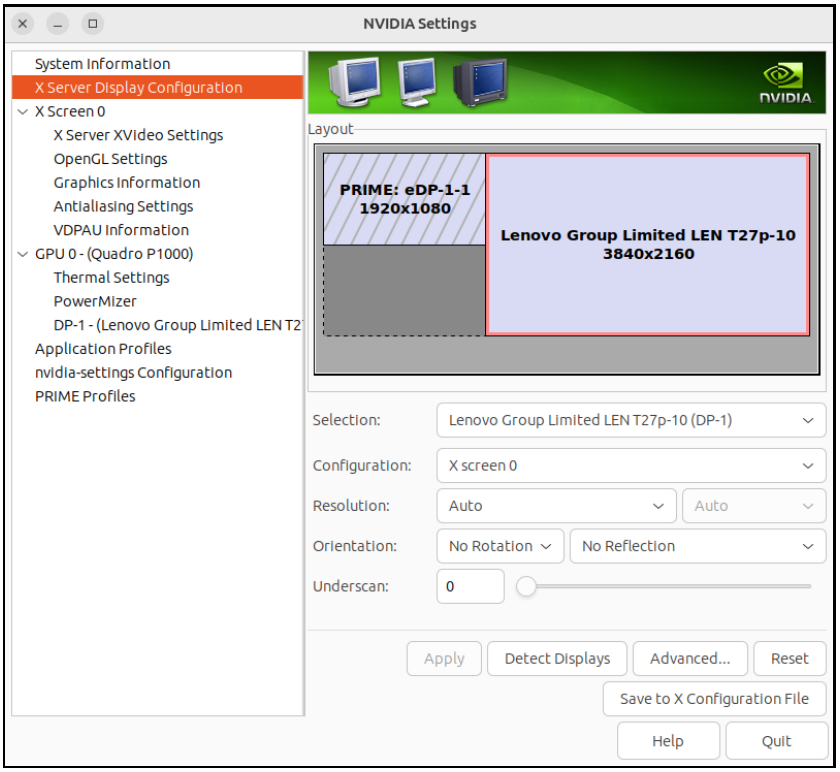


Figure 17.4 NVIDIA Driver Configuration

Undetectable Screens

If you have explicitly set the screen settings of GNOME or any other desktop system to use only one monitor, `nvidia-settings` will not display the other monitors! In the desktop settings, select a mode that uses all monitors (e.g., **Screen Mirroring**).

If this doesn't help or if external monitors have also disappeared in the desktop settings program, you should take a look at the `/lib/modprobe.d/nvidia-kms.conf` file. There may be an `modeset=1` entry. Replace it with `modeset=0`, then restart your computer.

PRIME PRIME is a technique to control which GPU is responsible for which screen on computers with multiple GPUs. In many notebooks that support either Intel or NVIDIA graphics (hybrid graphics, brand name Optimus), the built-in screen can be controlled either by the Intel CPU or by the NVIDIA GPU. The outputs for external displays, on the other hand, are mostly unalterably connected to the NVIDIA GPU.

In Figure 17.4, the internal screen is marked as the PRIME display. Because the Intel CPU is responsible for this, its settings cannot be changed by `nvidia-settings`. However, some notebooks allow you to set the BIOS/EFI to disable the hybrid system and only use the NVIDIA GPU. This gives the NVIDIA GPU full control over the built-in screen as well; it is then displayed as an ordinary screen in `nvidia-settings`.

prime-select (Ubuntu)

If the `nvidia-prime` package is installed on Ubuntu together with the NVIDIA drivers, then you can use the `prime-select` script included in it to switch the active GPU at runtime. `prime-select query` determines the current state:

```
user$ prime-select query
on-demand
```

The Intel GPU is usually used in `on-demand` mode. However, individual programs can be started via the **With Dedicated Graphics Card** context menu option. Theoretically, it should thus be possible to combine the best of both worlds: the energy-saving Intel GPU and the powerful NVIDIA GPU for demanding tasks. In my tests, however, this only worked moderately well.

`prime-select intel` switches to the Intel CPU. The setting will take effect only with the next login. External monitors can then no longer be used. In return, the notebook's energy consumption is reduced by several watts, which increases the battery life by one to two hours depending on the usage—ideal for a longer train ride without a power supply!

```
user$ sudo prime-select intel
```

Similarly, `prime-select nvidia` activates the constant use of the NVIDIA GPU for all programs. In stationary operation, this is the best setting.

nvidia-prime-select

In many distributions, the `nvidia-prime` package is not available. In most cases, the `nvidia-prime-select script` then works, which you can download and install from GitHub as follows:

```
user$ git clone https://github.com/wildtruc/nvidia-prime-select.git
user$ cd nvidia-prime-select
user$ sudo make install
```

After the installation, you need to perform the GPU change using `nvidia-prime-select intel` or `nvidia-prime-select nvidia`. Note that there have been no updates for the project in four years now.

17.4 Determining the Status of the Graphics System

It isn't easy to determine which drivers and components are active in a currently running graphics system.

X or Wayland?

A look at the process list reveals whether X (more precisely, the X server with the program name `Xorg`) or Wayland is running:

```
user$ ps ax | egrep -i "xorg|wayland"
   1516 tty2  ... /usr/libexec/gdm-wayland-session /usr/bin/gnome-
session
   2226 ?         /usr/bin/Xwayland :0 -rootless -noreset -accessx
-core ...
```

However, you must be careful when interpreting the result. There may be multiple instances of the graphics system running to display the login box as well as to display the desktop system for the currently logged-in user.

Some distributions run the display manager responsible for the login with Wayland, but then run the desktop environment with X.

The `loginctl` command, which outputs all sessions under the control of systemd, provides certainty:

```
user$ loginctl
SSESSION  UID USER   SEAT  TTY
      2 1000 kofler seat0 tty2
      6 1000 kofler       pts/1
```

In the second step, you can then determine details about a session, including whether the session uses X, Wayland, or just a teletypewriter (TTY) interface, as is the case with SSH logins or working in a text console:

```
root# loginctl show-session 2 | grep Type
Type=wayland
```

> **GNOME System Settings**
>
> If you work on GNOME, a look at the system settings is all it takes to recognize the active graphics system. The **Info** module announces whether the graphics system uses X or Wayland, under the somewhat ambiguous **Window Manager** item.

X server version

If your distribution uses X, you can determine the version of the X server by using the following command. The command must be executed in an SSH session or in a text console. The X.org server version 1.21 is installed on the computer in this example:

```
user$ sudo X -showconfig
X.Org X Server  1.21.1.8
X Protocol Version 11, Revision 0
...
```

An alternative approach is provided by the `xdpyinfo` command. The command must be executed in a terminal window within the graphics system. Thanks to Wayland's X compatibility, the command works equally well on X and on Wayland:

```
user$ xdpyinfo | grep 'X.Org version'
X.Org version: 23.1.1
```

Graphics card and driver

To find out which graphics card or graphics adapter you have and by which kernel driver it is controlled, you must analyze the output of the `lspci` command. The following lines show the results of two computers. One was equipped with an Intel CPU with integrated graphics unit, the other with an NVIDIA graphics card:

```
user$ lspci -k | egrep -A3 'VGA|3D|Display'
00:02.0 VGA compatible controller: Intel Corporation CoffeeLake-
S GT2
    [UHD Graphics 630]
  DeviceName: Onboard - Video
  Subsystem: ASUSTeK Computer Inc. Device 8694
  Kernel driver in use: i915
user$ lspci -k | egrep -A3 'VGA|3D|Display'
01:00.0 VGA compatible controller: NVIDIA Corporation GP107GLM
    [Quadro P1000 Mobile] (rev a1)
  Subsystem: Lenovo GP107GLM [Quadro P1000 Mobile]
  Kernel driver in use: nvidia
  Kernel modules: nvidiafb, nouveau, nvidia_drm, nvidia
```

3D functions (Mesa)

You can check the 3D functions of your system using the `glxinfo` program, which is hidden in the `mesa-utils`, `glx-utils`, or `Mesa-demo-x` package, depending on the distribution. The program provides a lot of detailed information about the running GLX system—that is, about the OpenGL 3D extensions of the X window system. `glxinfo` can also be run if you use Wayland.

You can use `grep` to filter out the crucial lines:

```
user$ glxinfo | grep render  (Intel driver, Wayland)
  direct rendering: Yes
  OpenGL renderer string: Mesa Intel(R) UHD Graphics 630 (CFL GT2)
```

```
user$ glxinfo | grep render    (NVIDIA driver, X)
  direct rendering: Yes
  OpenGL renderer string: Quadro P1000/PCIe/SSE2
```

If, on the other hand, no 3D-accelerated driver is running (e.g., in virtual machines), the output will look like one of the three examples ahead. In the first example, no 3D functions are available at all. In the second case, they are simulated by software using the Mesa library, and in the third case using the `llvmpipe` library. In the case of the `llvmpipe` library, the CPU takes care of the 3D functions. Although this increases the CPU's computing load considerably, it enables the use of 3D functions even without a proper 3D graphics driver:

```
user$ glxinfo | grep render
  Xlib: extension "GLX" missing on display ":0.0"
user$ glxinfo | grep render
  direct rendering: No
  OpenGL renderer string: Mesa GLX Indirect
user$ glxinfo | grep render
  direct rendering: Yes
  OpenGL renderer string: llvmpipe (LLVM 11.0.0, 256 bits)
```

inxi Instead of installing various tools and using all sorts of tricks to find the key data of the graphics system, you can also simply consult the `inxi` command presented in Chapter 14, Section 14.8. This computer with two monitors runs X with the proprietary NVIDIA driver:

```
user$ inxi -G
Device-1: Intel CoffeeLake-H GT2 [UHD Graphics 630] driver: i915
v: kernel
Device-2: NVIDIA GP107GLM [Quadro P1000 Mobile] driver: nvidia
v: 525.105.17
Device-3: Acer Integrated Camera type: USB driver: uvcvideo
Display: x11 server: X.Org v: 1.21.1.7 with: Xwayland v: 22.1.8 driver:
X:
  loaded: modesetting,nvidia unloaded: fbdev,nouveau,vesa dri: iris
  gpu: i915,nvidia,nvidia-nvswitch resolution: 1: 3840x2160~60Hz
                                              2: 1920x1080~60Hz
API: OpenGL v: 4.6.0 NVIDIA 525.105.17 renderer: Quadro P1000/PCIe/SSE2
```

Logging When X is started, numerous messages, warnings, and possibly also error messages are logged. It depends on the configuration of your distribution where this data is stored:

- `/var/log/Xorg.0.log` contains the X log for the display manager.

- `.local/share/xorg/Xorg.0.log` or `.local/share/xorg/Xorg.1.log` contains the log for the active session.
- In some circumstances, the `journalctl` command also provides the logging messages. However, because `journalctl` is a general logging tool, the program does not only provide X-specific messages. To filter out the relevant output, you can try the following command:

  ```
  user$ journalctl --user | grep gdm | tail -n 100
  ```

  This provides the last 100 logging messages of the display manager, `gdm`. (If you do not use GNOME, you must first determine the name of your distribution's display manager. Usually, it's `sddm` or `lightdm`.)

The X start log contains detailed information about which configuration file was used, which modules were loaded, which problems occurred, which graphics modes were discarded and for what reasons, and so on. Entries within the logging file are identified by the following codes:

- (**), setting from the configuration file
- (++), setting from the command line
- (==), X default setting
- (--), setting resulting from detected hardware
- (!!), note
- (II), note
- (WW), warning
- (EE), error

If there are multiple X logging files in `/var/log/`, you must look for the most recent file. Due to the wealth of information in `Xorg.0.log`, the search for really relevant data is unfortunately like the proverbial search for a needle in a haystack. If necessary, you can simply send the entire logging file to someone more familiar with it or post it to a support forum.

If the file is missing, the graphics system probably uses Wayland. Unfortunately, the other extreme applies there: Wayland dispenses with almost all logging. This is mainly due to the fact that Wayland does not have its own process that is comparable to the X server. Wayland only defines logs that are implemented in libraries. If errors occur there, error messages may be passed to the syslog service. You can extract these messages from the journal or the central syslog log using `grep`:

```
root# journalctl | grep -i wayland | tail -n 100
```

Tips for troubleshooting Wayland problems can be found on the following page: *https://fedoraproject.org/wiki/How_to_debug_Wayland_problems*.

17.5 Starting the Graphics System

This section summarizes some information about starting and stopping the graphics system. You only need to deal with this if the automatic startup doesn't work when booting the computer or if you want to intervene in the process manually.

systemd

For desktop distributions, the graphics system is started by default. The systemd target `graphical` is responsible for this (see Chapter 20, Section 20.1). To switch from graphics mode to text mode or back to graphics mode, you'll want to run `systemctl isolate`. The command becomes effective immediately. When you switch to text mode, all currently running programs are terminated immediately. You should therefore save all open files beforehand! If only a black screen is visible after switching to text mode, you must use Alt+F2 or Alt+F3 to switch to another text console:

```
root# systemctl isolate multi-user      (enable text mode)
root# systemctl isolate graphical       (enable graphics mode)
```

`systemctl isolate` applies immediately, but only until the next reboot of the computer. If you want to switch permanently between text and graphics mode, you need to change the default target of ystem:

```
root# systemctl set-default multi-user  (start in text mode in future)
root# systemctl set-default graphical   (start in graphics mode in future)
```

The display manager service (in most cases, `gdm`) must also be activated so that the display manager appears with the login box when the graphics system is started. In multiuser mode, this should be the case by default:

```
root# systemctl status gdm
gdm.service - GNOME Display Manager
     Loaded: loaded (/lib/systemd/system/gdm.service; static)
     Active: active (running) since Tue 2024-04-02
```

Only a few distributions—for example, Arch Linux after a manual installation of GNOME—require you to explicitly enable the service:

```
root# systemctl enable --now gdm
```

17.5.1 Wayland or X?

Current Linux distributions are usually set up in dual mode: both the X and Wayland graphics systems are available for selection. But how can you influence which graphics system is used? For many distributions, you select the preferred mode when you log in using the gear menu or a similar mechanism (see Figure 17.5).

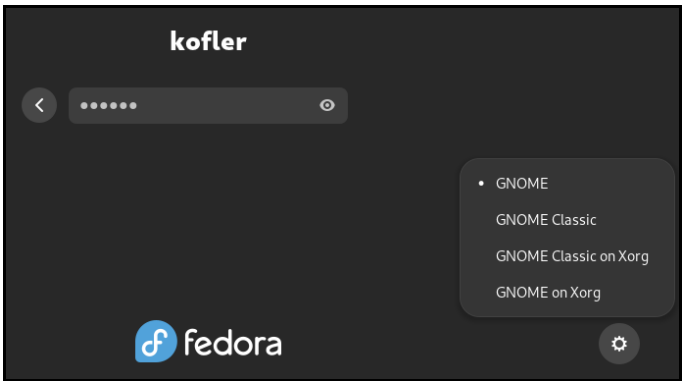


Figure 17.5 GNOME Display Manager: Choose GNOME or GNOME Classic, on either Wayland or X

The display manager responsible for the login remembers your decision until the next time you select another entry. If Wayland is not available, the corresponding options will not be displayed at all.

17.5.2 The Role of the Display Manager

What actually happens when the init system activates the graphics system—that is, when systemd starts the `graphical` target in current distributions?

For the time being, a so-called display manager is started. Although this program itself runs in graphics mode, it only provides a login box. Only after successful login does the display manager start the desktop system—on many distributions, either GNOME or KDE.

As is usual for Linux, there is not just one implementation of the display manager, but several. The most widely used one is the *GNOME Display Manager* (the `gdm` program; see Figure 17.5). Popular alternatives include the *Simple Desktop Display Manager* (`sddm`, used on KDE) and the *Light Display Manager* (`lightdm`) used by Raspberry Pi OS. The minimalist `xdm` program, the original display manager of the X window system, is now only of historical significance.

Accordingly, different service files are also responsible for systemd starting each display manager:

```
/lib/systemd/system/gdm.service                (Debian/Ubuntu, Fedora, RHEL)
/lib/systemd/system/sddm.service               (Kubuntu, Neon)
/lib/systemd/system/lightdm.service            (Raspberry Pi OS)
/usr/lib/systemd/system/display-manager.service (SUSE)
```

17.5.3 Configuring the Display Manager

gdm

The `gdm` configuration files are located in `/etc/gdm` or `/etc/gdm3`. Among other things, they control which programs, commands, and scripts are to be used for various functions of the display manager. If for some reason you want to disable Wayland, you need to add the following lines to one of the configuration files:

```
# in /etc/gdm/custom.conf bzw. /etc/gdm3/custom.conf
[daemon]
# Disable Wayland
WaylandEnable=false
```

lightdm

The configuration of `lightdm` is done by two minimalistic files in `/etc/lightdm`. Changes to these files are rarely required. You can use the `allow-guest=false` statement in `lightdm.conf` to disable guest sessions on Ubuntu.

sddm

`sddm` is configured by the `/etc/sddm.conf` file. Instead of modifying the file directly, you can use the **Startup and Shutdown • Login Screen (SSDM)** dialog sheet of the KDE system settings.

***.desktop files**

If you have multiple desktop systems installed in parallel (i.e., KDE *and* GNOME), you can select which desktop or window manager to start in a menu when you log into the display manager. The data for this menu is located in `*.desktop` files in the `/usr/share/xsession` directory (keyword `SessionDesktopDir` in the `gdm` configuration) on most distributions.

For example, the desktop file to start GNOME contains the following lines:

```
[Desktop Entry]
Name=GNOME
Comment=This session logs you into GNOME
TryExec=/usr/bin/gnome-session
...
```

17.5.4 Automatic Login

On desktop systems, it's sometimes preferable for the standard user to be logged in automatically when the computer is started. This is convenient, but of course a security risk. On many distributions, you can configure auto login in the system settings; on GNOME systems and on Ubuntu, you can do so in user management. The following paragraphs describe the manual configuration.

If your distribution uses gmd as the display manager, you should add the following lines to the [daemon] section of custom.conf: **gdm**

```
# file /etc/gdm3/custom.conf or /etc/gdm3/daemon.conf
...
[daemon]
  AutomaticLoginEnable=true
  AutomaticLogin=<loginname>
```

For lightdm, the autologin-user variable in lightdm.conf controls the automatic login: **lightdm**

```
# file /etc/lightdm/lightdm.conf
[SeatDefaults]
autologin-user=<loginname>
...
```

openSUSE provides its own configuration files for automatic login. Direct changes in KDE or GNOME are not recommended because SuSEconfig will overwrite the settings at the next opportunity. Instead, you should modify the /etc/sysconfig/displaymanager file. To disable the autologin function, you need to assign an empty string to the AUTOLOGIN variable. The changes do not become valid until you execute the SuSEconfig command: **openSUSE**

```
# /etc/sysconfig/displaymanager
...
DISPLAYMANAGER_AUTOLOGIN="username"
```

As an alternative, you can of course perform the configuration in YaST. To do this, you need to run the **Options for Experts • Login Settings** command in the **Users and Groups** module.

17.5.5 Monitor Configuration for gdm

Many distributions use GNOME as the desktop and gdm as the display manager. Unfortunately, in multimonitor setups, gdm displays the login box only on the first screen, and it's often difficult to predict which screen will be preferred—and it's not necessarily the screen you would expect.

KDE or sddm acts more intelligently in this respect and mirrors the login on all screens. To achieve this behavior on GNOME as well, you must temporarily set your desktop to **Screen Mirroring** mode as well. Then copy the .config/monitors.xml file to the gdm configuration directory:

```
user$ sudo cp ~/.config/monitors.xml ~gdm/.config/monitors.xml
user$ sudo chown gdm:gdm ~gdm/.config/monitors.xml
```

However, the file is only taken into account if X is active as the graphics system. You can force this if necessary by making the following change to `/etc/gdm3/custom.conf`:

```
# change in /etc/gdm3/custom.conf or /etc/gdm3/daemon.conf
...
WaylandEnable=false
```

Chapter 18
File System Administration

This chapter describes various facets of file system administration. It is aimed at advanced Linux users and covers the following topics:

- **How everything is connected**
  This section provides an initial overview of how various aspects of the Linux file system interact.
- **Formatting and using USB media**
  For all impatient users, this section summarizes how you can format a USB flash drive or SD card at the command level or how to integrate an external hard disk permanently into the directory tree. The relevant principles are then explained later in the chapter.
- **Device names**
  Within Linux, hard disks and other data media and their partitions are addressed via device files such as `/dev/sdc3`. This section summarizes the schema for terminology and numbering.
- **Partitioning the hard disk/SSD**
  Partitioning a hard disk, SSD, or virtual disk (in virtualization systems or in the cloud) represents a central part of the installation of Linux. However, sometimes it's necessary to add a new partition when you run Linux. Depending on which partitioning type applies to the hard disk (MBR or GPT), you need to use different tools to change the partitioning.
- **File system types**
  Only a few operating systems support as many file system types as Linux. This section summarizes the most important variants.
- **File system management**
  Here you will learn how you can manually insert individual data partitions into the file system (`mount`) and how to automate this process (`/etc/fstab`).
- **Linux and Windows file systems**
  Several sections provide tips and information on using the `ext4`, `btrfs`, `xfs`, `vfat`, and `ntfs` file systems.
- **Swap partitions or files**
  If Linux has too little memory to run the programs, it stores parts of the memory in swap partitions or in swap files.
- **RAID**
  RAID enables you to link the partitions of multiple hard disks in order to

achieve a more reliable and/or faster overall system. This section briefly reviews the basic principles of RAID.

- **LVM**
  The Logical Volume Manager enables more flexible management of partitions. For example, you can use LVM to combine partitions of multiple hard disks into one virtual partition, change the size of partitions during operation, and more.
- **SMART**
  Self-Monitoring, Analysis, and Reporting Technology (SMART) makes it possible to collect statistical data during the operation of hard disks and in this way detect impending reliability issues even before data loss occurs.
- **SSD TRIM**
  The SSD TRIM function, which is supported by all current solid-state disks, allows the operating system to tell the SSD which memory blocks of the file system are unused after a file has been deleted. The SSD can then optimize the internal use of the memory cells.
- **Encrypted file systems**
  If you want to prevent unauthorized persons from reading your data—for example, after a computer theft—then you must encrypt your files or file systems. Linux provides different methods for this, the most popular being based on the `dm_crypt` kernel module.

There is, of course, more that could be said or written about the administration of file systems. There's not that much room here, though. Instead, a few cross-references will have to suffice here:

- **Using the file system**
  Commands for copying files or creating backups, background information on the access rights of files, and so on have already been presented in this book in Chapter 9.
- **Network file systems**
  On Linux, you can also integrate directories of other computers into your directory tree. In this book, I discuss the CIFS file system (Windows/Samba; see Chapter 27, Section 27.7).
- **Disc quotas**
  This is a system that controls how much space individual users are allowed to take up on the hard disk. If the limit is exceeded, no new files can be created. Disk quotas are practical when a computer provides storage space for many users (school children, students, etc.). For a good introduction, see *https://linuxconfig.org/how-to-use-disk-quota-on-linux-with-examples*.

- **Cluster file systems**
  Cluster file systems or global file systems combine data from multiple computers into a virtual file system. This means that huge data storages can be created and used by multiple computers in parallel.

  Once again, Linux provides different methods to create such file systems, such as via Oracle Cluster Filesystem (OCFS; *https://oss.oracle.com/projects/ocfs2*), Global Filesystem (GFS; *https://sourceware.org/cluster/gfs*), or Ceph (*https://ceph.io*).

18.1 How Everything Is Connected

The many aspects of managing the file system can sometimes be confusing. This section attempts to briefly outline the most important relationships. To keep the text clear, I will limit myself here to built-in hard disks/SSDs and standard Linux file systems. DVD drives, external data media, LVM and RAID systems, and the rest are left out for the time being; details will follow in the subsequent sections.

Hard disks and SSDs

On Linux, device files are assigned to the built-in hard disks and SSDs. All common distributions use `/dev/sda` for the first disk, `/dev/sdb` for the second one, and so on. Deviating from this, PCIe SSDs are given device names in the following format: `/dev/nvme0n1`.

If you use Linux in virtualization systems or in the cloud, the data media is virtual and is often given device names such as `/dev/vda` or `/dev/vdb`. From the perspective of this chapter, this doesn't change much. But while the size of an SSD or hard disk is unchangeable, the capacity of cloud data storage can be changed at the click of a mouse or by upgrading—possibly even during operation.

Partitions

To accommodate multiple, independent file systems on a data medium, it must be divided into sections (partitions). Device files are also assigned to the partitions, such as `/dev/sda1` for the first partition of the first hard disk. The device naming convention for partitions is summarized in detail in Section 18.3.

System partition

When Linux is started, the kernel first accesses the system partition (root partition). To enable the kernel to find the system partition, its device name or, better, the Universal Unique Identifier (UUID) of the file system it contains is passed in a kernel parameter. The GRUB configuration is responsible for this.

Other partitions

In addition to the system partition, which is absolutely necessary, there may be other partitions whose file systems are to be taken into account

when Linux is started. In almost all distributions, these files are listed in the `/etc/fstab` file. This file must in turn be located in the system partition. It is analyzed as part of the init process.

Consistency test/repair

When partitions are included in the directory tree, the consistency of the file systems is automatically checked. If the computer last crashed or was not shut down properly due to a power failure, the file system is automatically repaired or other safety measures are taken to prevent consistency errors or damage. A corresponding consistency test can also be carried out automatically after a certain period of use. The details of this process depend on the distribution, the file system type, and the individual configuration.

Directory tree instead of drive letters

While it is common on Windows to address separate file systems via drive letters (*C:*, *D:*, etc.), on Linux all file systems are combined in a directory tree. The system partition is accessed via the root directory, /. The starting point of all other file systems can vary depending on the distribution and configuration. Typical subdirectories include `/mnt`, `/media`, and `/run`—for example, `/media/dvd` or `/run/media/username/dvdname`.

Adding file systems

It's possible to add further file systems to the directory tree during operation or to remove them again. This is usually done automatically when an external data medium (such as a USB flash drive) is plugged in. If this automatic process doesn't work or if it has been deliberately deactivated, the `root` user can use the `mount` and `umount` commands to manually mount or unmount file systems. The only constant is the system partition: it cannot be removed from the file system during operation. This is only possible when the computer is shut down.

File system types

Linux supports many file system types. The system partition must be in a Linux file system (e.g., `ext4`, `btrfs`, or `xfs`). The choice is even greater for the remaining partitions. Windows, Unix, or Apple file systems, for example, are also possible.

18.2 Formatting and Using USB Media

As a foretaste, so to speak, of the following sections, I will summarize here how to deal with external data media on Linux at the command level. This section is intended for Linux users who already have some experience working in the terminal and do not always use a desktop system such as KDE or GNOME. Details and basic information on the commands presented here will follow later in the chapter.

18.2.1 Formatting a USB Flash Drive or SD Card

There are often times when you want to quickly format a USB flash drive or SD card on Linux so that the data medium can later be used on a Windows notebook computer or in the camera. The procedure described here also applies to external hard disks.

Determining the device

First you need to determine the device name of the USB flash drive using `lsblk`. The `SIZE` and `RM` (*removable media*) columns are important for identifying the correct device:

```
root# lsblk
NAME            MAJ:MIN  RM   SIZE RO TYPE MOUNTPOINT
nvme0n1         259:0    0 476.9G  0 disk
  nvme0n1p1     259:1    0     2G  0 part  /boot/efi
  nvme0n1p2     259:2    0   128G  0 part  /
sda               8:0    1   1.8T  1 disk
  sda1            8:1    1   1.7T  1 part  /run/media/kofler/backup
sdb               8:32   1   7.4G  1 disk
  sdb1            8:33   1   1.9G  1 part  /run/media/kofler/No Title
root# umount /dev/sdb1
```

In this example, there is an internal SSD with a Linux installation, an external SSD as a backup medium, and a USB flash drive of just under 8 GiB. This is to be formatted. The correct device name is therefore `/dev/sdb`.

If the data medium already contains file systems and these are active (`MOUNTPOINT` column), you must remove them from the directory tree. In this example, this applies to the `/run/media/kofler/No Title` directory.

Partitioning and formatting

Sometimes it's also necessary to repartition the USB flash drive or SD card. In the following listing, this is done by using the two `parted` commands. Note that the following `parted` command deletes the entire data medium! Finally, `mkfs.vfat` sets up the new file system, which is given the name (volume label) `PHOTOS`:

```
root# parted /dev/sdb mklabel msdos
  Warning: The existing partition table and all data
  on /dev/sdb will be deleted. Do you want to continue? yes
root# parted /dev/sdb 'mkpart primary fat32 1mib -1mib'
root# mkfs.vfat -F 32 -n PHOTOS /dev/sdb1
```

Depending on the intended use, it's better to set up an NTFS or a Linux file system instead of the VFAT file system. To do this, you can proceed as follows:

```
root# mkfs.ntfs --fast /dev/sdb1   (NTFS)
root# mkfs.ext4        /dev/sdb1   (Linux file system)
```

Before you can run `mkfs.ntfs`, you may need to install the `ntfsprogs` package.

18.2.2 Mounting USB Media Manually

On KDE, GNOME, or a comparable desktop system, USB media are automatically mounted in the directory tree after being plugged in; a file manager window usually also appears on the screen so that you can access the files. It's also possible that your drive is initially only displayed in Nautilus or the KDE file manager and is not mounted until you click it. The first access to the files then takes a short time; in the case of an external hard disk, you can hear its small motor start up.

Behind the scenes, a `mount` command is executed—and you have to execute this command manually if you do not use a desktop system. To do this, you must first determine the correct device name using `lsblk`. Most USB data media are formatted in such a way that the file system is located in the first partition (e.g., `/dev/sdb1`).

Sometimes the entire data medium is used without partitioning, in which case the device name would be, for example, `/dev/sdb`. Finally, a data medium may contain more than one partition and therefore more than one file system. This is particularly common with hard disks.

As soon as you know which device you want to access, you can try to execute the following commands:

```
root# mkdir /media/data
root# mount /dev/sdb1 /media/data
```

`mount` recognizes the file system itself and uses the default settings for this file system. These settings are sufficient for `root` to access the files:

```
root# mount  /dev/sdb1 /media/data
```

In Section 18.8, you will learn about some other options that allow users other than `root` to modify files on a Windows file system.

18.2.3 Mounting an External Hard Disk Automatically

Sometimes external hard disks are constantly connected to the computer—for example, as a backup location or as additional storage because the internal hard disk is too small. If you use a desktop system such as KDE or

GNOME, the system automatically mounts the disk using directories such as /media/filename or /run/media/loginname/filename. In this case, the information presented here is not relevant to you.

But if you want to access the data medium via a different directory or work without a desktop system, you must modify /etc/fstab. The aim is for Linux to permanently integrate the data medium into the directory tree during the boot process. You need to know three things to configure fstab correctly:

- Which file system is on the hard disk?
- What UUID number does the file system use?
- What is the UID number of the user who is to access the data?

Determining the file system type and UUID

To answer the first two questions, you need to connect the USB cable of the hard disk to your computer, determine the current device name using lsblk, and determine the remaining data via blkid. In the following example, the data media contains a VFAT file system. The UUID is F5EC-031F:

```
root# lsblk
...
sdb               8:16   1   7.4G  0 disk
  sdb1            8:17   1   7.4G  0 part
root# blkid /dev/sdb1
/dev/sdb1: LABEL="MYDISK" UUID="F5EC-031F" TYPE="vfat" PARTUUID=
"fac67ace-01"
```

UID

The user identification (UID) number of the first user set up on the Linux computer is usually 1000. The UIDs of other users can be found in the third column in /etc/passwd if required.

Setting up the directory

The data medium is to be used in future via the /media/data directory:

```
root# mkdir /media/data
```

fstab settings for vfat

To ensure that Linux mounts the data carrier there during the boot process, you should add the following data to /etc/fstab using an editor. It's important that all six columns be entered in one line. This isn't possible here for space reasons. Also note that the options introduced with uid=1000 must be separated by commas (not by spaces!).

```
UUID=F5EC-031F /media/data vfat uid=1000,gid=1000,fmask=0022,\
    dmask=0022,flush,utf8,noexec,nofail 0 0
```

These settings ensure that the user with UID 1000 can read and change files and directories. In most distributions, this is the first user that is set up. If the hard disk is accessed by a script that runs with root permissions, you can

omit the `uid`, `gid`, `fmask`, and `dmask` options; then only `root` has write permissions. A detailed description of these four options can be found in `man fstab`.

All other users have read access to the data medium. For security reasons, no one is allowed to execute the programs on the data medium (`noexec`). If the hard disk is not connected to the computer during the next boot process, the boot process will continue without error. If the hard disk is plugged in later, however, `mount` must be run manually:

```
root# mount /media/data
```

fstab settings for ntfs

Depending on the file system type, the line in `/etc/fstab` must be adjusted. The following settings apply to an NTFS file system:

```
UUID=713052474ADB2774 /media/data ntfs \
  uid=1000,gid=1000,fmask=0022,dmask=0022,flush,nofail,noexec 0 0
```

fstab settings for ext4

For a Linux file system such as `ext4`, the same access rules apply as for local file systems. The `uid`, `gid`, `fmask`, and `dmask` options are therefore omitted:

```
UUID=e593f718-27a4-475a-be4a-bec358d38796 /media/data ext4 \
  nofail,noexec 0 0
```

18.3 Device Names for Hard Disks and Other Data Media

SATA and PCIe are currently the usual standards for connecting a computer to its data media. Table 18.1 summarizes the meaning of these and some other abbreviations.

Abbreviation	Meaning
ATA	Advanced Technology Attachment (old interface for connecting hard disks)
PCIe	Peripheral Component Interconnect Express (fast data bus for SSDs)
SATA	Serial ATA (ATA variant with serial data transfer)
SCSI	Small Computer System Interface (formerly popular interface for server hard disks)

Table 18.1 Glossary

Internal kernel details

Within Linux, internal and external hard disks, SSDs, and CD and DVD drives are accessed via device files. These are special files that serve as an interface between Linux and the hardware.

You only need these device files for administration purposes—that is, if you want to change the partitioning of a hard disk or integrate a specific partition into the file system. In regular operation, you access the entire file system via directories. Here, / denotes the start of the file system. Individual partitions can be included at any location—an additional Linux partition named `/data`, for example, or a Windows partition named `/media/win`.

In the kernel, the SCSI driver is responsible for internal and external data media that are connected via the IDE, SATA, SCSI, USB, or Firewire bus systems. The *SCSI* driver name has a history: originally, the SCSI driver was intended only for SCSI devices, but gradually more and more bus systems were integrated into this driver without its name changing.

Device names

SATA and USB data media are named using `/dev/sdXN`. The storage media are therefore named `/dev/sda`, `/dev/sdb`, and so on in sequence (see Table 18.2).

Device	Meaning
`/dev/sda`	First hard disk/SSD (SATA/USB)
`/dev/sdb`	Second hard disk/SSD (SATA/USB)
...	
`/dev/nvme0n1`	First PCIe SSD
`/dev/nvme1n1`	Second PCIe SSD
...	
`/dev/scd0` or `/dev/sr0`	CD/DVD drive
`/dev/mmcblk0`	SD card (Raspberry Pi)

Table 18.2 Device Names of Data Media

For SATA devices, all channels used are connected in sequence with a letter. Modern mainboards provide up to eight channels for this. For example, if two hard disks are connected to SATA channels 1 and 3, they are given the device names `/dev/sda` and `/dev/sdb`. If a third hard disk is later connected to channel 2, the device name of the second hard disk changes from `/dev/sdb` to `/dev/sdc`.

External USB devices are treated like SATA devices. The order in which the devices are connected is decisive for the assignment of the letters. CD and DVD drives are given their own device names, which are `/dev/scdN` or `/dev/srN`, depending on the distribution.

PCIe SSDs are a special case, as are SD cards on some devices (especially minicomputers such as the Raspberry Pi). They are given rather cumbersome device names such as `/dev/nvme0n1` or `/dev/mmcblk0`.

> **Caution: Inconsistent PCIe Numbering**
>
> On a computer with *one* fast PCIe SSD, the device name is always `/dev/nvme0n1`. However, if there are multiple SSDs, it depends more or less on chance which SSD is addressed as `/dev/nvme0n1` and which as `/dev/nvme1n1`. This can change with every boot process—depending on which SSD is faster during initialization! Under no circumstances should you rely on the device names being consistent: you must use labels or UUIDs in the `/etc/fstab` file. The following SUSE support page points out that the inconsistent naming of the devices is not an error, but the expected behavior: *https://www.suse.com/support/kb/doc/?id=000019309*.

Virtual disks (virtio)

If Linux is executed in a virtual machine and the *virtio* driver is used, the kernel addresses the virtual hard disks via device names such as `/dev/vda`, `/dev/vdb`, and so on. The virtio driver enables particularly efficient communication between the virtualization system and the kernel in the virtual machine. The KVM and Xen virtualization systems support virtio as a standard feature.

Partition numbers (MBR)

The numbering structure for partitions depends on how the hard disk is divided up. Two partitioning variants are currently possible: the classic MBR method, in which the partitioning table is located in the master boot record (MBR); and the newer GUID partition tables (GPTs), which are often used for internal hard disks and SSDs (see Section 18.4).

Device	Meaning
`/dev/sda`	The first SATA disk
`/dev/sda1`	The first primary partition of this volume
`/dev/sdd3`	The third primary partition of the fourth SATA volume
`/dev/sdd5`	The first logical partition of the fourth SATA volume
`/dev/sdd15`	The eleventh logical partition of the fourth SATA volume

Table 18.3 Examples of Partition Numbering (MBR)

In MBR partitioning, digits 1 to 4 are reserved for primary or extended partitions and digits from 5 on for logical partitions within the extended partitions. For this reason, it's quite common for there to be gaps in the numbering. For example, if the hard disk has one primary, one extended,

and three logical partitions, these will have the numbers 1, 2, 5, 6, and 7. Table 18.3 contains some examples. The number of partitions per disk is limited to 64.

Partition numbers (GPT)

The numbering of hard disks with GPT is much simpler. There is no need to differentiate between primary, extended, and logical partitions; the partitions are simply numbered consecutively, with up to 128 partitions permitted.

Unconventional Numbering

It's possible for the physical order of the partitions to differ from the numbering! Let's assume that three partitions with 1 TiB each have been created on a hard disk with 3 TiB (`/dev/sda1` to `/dev/sda3`). The middle partition is then deleted. Two new partitions with 500 GiB each are now created in the free area. These two partitions are given the device names `/dev/sda2` and `/dev/sda4`! This special case is not possible with MBR partitioning because only one partition can be inserted between `/dev/sda1` and `/dev/sda3`.

lsblk

An extremely useful and valuable tool on computers with multiple disks is `lsblk`. It provides a tree-like list with the device names of all disks and partitions, including their use or `mount` point. The following example was created on a notebook computer. It contains two PCIe SSDs, and an external USB hard disk is also connected for backups. One SSD with 477 GiB contains multiple partitions that are not currently in use. The other SSD contains a 1.8 TiB encrypted partition, which is used for LVM:

```
root# lsblk
NAME                    MAJ:MIN RM   SIZE RO TYPE  MOUNTPOINTS
sda                       8:0    0   1.8T  0 disk
  sda1                    8:1    0   1.7T  0 part  /run/media/kofler/
p1-backup2
nvme0n1                 259:0    0 476.9G  0 disk
  nvme0n1p1             259:1    0     2G  0 part
  nvme0n1p2             259:2    0   128G  0 part
  nvme0n1p3             259:3    0   128G  0 part
nvme1n1                 259:4    0   1.8T  0 disk
  nvme1n1p1             259:5    0     2G  0 part  /boot
  nvme1n1p2             259:6    0   1.8T  0 part
    cryptlvm            254:0    0   1.8T  0 crypt
      vgcrypt-archroot  254:1    0   200G  0 lvm   /
      vgcrypt-archhome  254:2    0   600G  0 lvm   /home
      vgcrypt-data      254:3    0   700G  0 lvm   /data
```

If `lsblk` displays `loop` devices, then these are not real data media, but images of the Ubuntu-specific Snap packages (see Chapter 16, Section 16.11).

Alternative device names

As I have already explained, it's very risky to rely on the device names of data media because they can change. To ensure that you can address individual devices or partitions uniformly despite varying device names (e.g., in a backup script), the `/dev/disk` directory contains additional links to all data media, which are organized according to various criteria:

- `/dev/disk/by-id/` uses IDs composed of the bus system, the device name, and a model or serial number.
- `/dev/disk/by-label/` uses the name given to the file system.
- `/dev/disk/by-path/` uses a pathname derived from the PCI interface, bus system, and partition number. *Caution:* If a USB device is connected to a different USB connector the next time, its path name will change!
- `/dev/disk/by-uuid/` uses the UUIDs of the file systems. UUIDs are unique ID numbers assigned to a file system during the formatting process. These UUIDs enable file systems to be identified even after a change to the hardware configuration.

The number of links in the `/dev/disk` directories varies. `/dev/disk/by-label` and `by-uuid`, for example, only contain links to partitions that are named or have a UUID. The `udev` system is responsible for automatically generating the links (see Chapter 9, Section 9.9). The following `ls` command shows some links on my notebook. To improve readability, I have indented the lines a little and removed the information on access rights:

```
user$ cd /dev/
user$ ls -l disk/by-id
  nvme-Samsung_...997Z         -> ../../nvme1n1
  nvme-Samsung_...997Z-part1   -> ../../nvme1n1p1
  nvme-Samsung_...997Z-part2   -> ../../nvme1n1p2
   ...
disk/by-path:
  pci-0000:71:00.0-nvme-1       -> ../../nvme1n1
  pci-0000:71:00.0-nvme-1-part1 -> ../../nvme1n1p1
  pci-0000:71:00.0-nvme-1-part2 -> ../../nvme1n1p2
  ...

disk/by-uuid:
  E4E5-DA46                             -> ../../nvme1n1p1
  44fbbcd5-8d8e-4978-9c32-ffcaa646554f -> ../../nvme1n1p2
  ...
```

18.4 Partitioning the Hard Disk or SSD

There is nothing mysterious about partitioning a hard disk or SSD. The aim is simply to divide the data medium into areas for different uses.

The problem, however, is that the basic principles of partitioning go back to the 1980s, when a hard disk with 50 MiB was *huge* and nobody could imagine a solid-state disk. Over the past decades, data storage technology has made almost unimaginable progress, while the fundamentals of partitioning have changed only very tentatively.

GPT is a truly modern standard for storing partitioning data and is now used across all operating systems on internal hard disks and SSDs. Volumes in virtual machines and external data media (especially USB flash drives and SD cards), on the other hand, often still use MBR partitioning, which dates back to DOS times.

> **Warning: Be Careful**
>
> Partitioning programs can destroy the contents of your entire hard disk or SSD! Read this section in its entirety before using partitioning tools! You must never change a partition that is currently in use—that is, whose file system is integrated or "mounted" in the directory tree!

Partitioning tools

During installation, most distributions offer easy-to-use partitioning tools. Only a few distributions also provide these tools during operation—for example, the **System • Partition** YaST module on SUSE. Otherwise, you have to run special partitioning tools for later changes. The two most important tools are the `parted` command and its `gparted` graphical user interface.

LVM

If you need to make frequent changes to the partitioning, you should definitely familiarize yourself with LVM (see Section 18.16). LVM inserts a virtual layer between the physical partitions of the hard disk and the partitions used for file systems and makes subsequent changes much easier.

Doing without partitions

Basically, it's possible to dispense with partitions and set up a file system directly on the data medium. This is totally unusual for real hardware. For cloud installations, however, it can make sense to set up the file system directly on the data medium. If the virtual data medium is later enlarged, you can then easily enlarge the file system using `resize2fs` or `xfs_growfs` without having to deal with partitions.

18.4.1 MBR or GPT?

There are two ways in which partitioning data can be saved: in the MBR or via GPT. The MBR method has been in use for decades but is only suitable for data media up to 2 TiB. The use of GPT is mandatory on Linux in two cases:

- For hard disks or SSDs larger than 2 TiB
- For hard disks or SSDs that are also used in parallel to boot Windows or macOS

Even if GPT is optional for Linux, there is no good reason to still use MBR today (except for USB flash drives or SD cards). I myself generally set up GPT on new hard disks and SSDs because I'm tired of messing around with primary, extended, and logical partitions.

The Partitioning System Is Difficult to Change at a Later Stage!

A later switch between MBR and GPT will result in the loss of all data, unless you use less common special tools. If necessary, you should take a look at the following two pages:

- *https://askubuntu.com/questions/84501*
- *https://rodsbooks.com/gdisk/mbr2gpt.html*

Initializing the hard disk/SSD

Current Linux distributions can handle MBR and GPT media. However, when it comes to setting up new partitioning, many installation programs do not offer a choice between MBR and GPT. On empty data media that are smaller than 2 TiB, MBR is often used without you being asked.

If you want to determine the partitioning type yourself, manual work is required, which is best carried out in a live system. Start the `parted` program in a terminal window with `root` permissions and execute the corresponding `mklabel` command. Enter the correct device name of a medium (not a partition) instead of `/dev/xxx`:

```
root# parted /dev/xxx
(parted) mklabel gpt        (for GPT)
(parted) mklabel msdos      (for MBR)
(parted) quit
```

Caution: When using `mklabel`, you will lose all data on the disk in question!

18.4.2 Basic Rules

Regardless of the tool you use, you need to observe a few basic rules:

- It's impossible to change the system partition during operation. You may have to boot the computer with a live system from a USB flash drive. You can find distributions for hard disk partitioning on the following pages:
  - *https://gparted.org/livecd.php*
  - *https://partedmagic.com*
  - *https://www.system-rescue.org*
- A partition can only be enlarged if there is free space behind the partition. You cannot move partitions.
- If you change the size of a partition, this does *not* automatically change the size of the file system it contains! This requires additional commands, which vary depending on the file system type.
- For speed reasons, all partitions must start at a multiple of 4 KiB. Today it has become common practice to use a multiple of 1 MiB.
- Finally, it's customary to leave the first MiB of the disk free so that there is room for the boot loader code. (On EFI systems, the boot loader ends up in its own partition anyway. However, virtual machines are rarely EFI-compatible.)

18.5 The parted Command

`parted` is the most important partitioning tool for Linux at the command level. It can handle both MBR and GPT partitions. Details on the many functions of `parted` can be found at *https://gnu.org/software/parted*.

fdisk: A Relic from the Past

As an alternative to `parted`, `fdisk` is still popular today. However, as `fdisk` is only suitable for data media with MBR partitions, I will not discuss `fdisk` in this book.

Start

When starting `parted`, you pass the device of the data medium as a parameter. `help` or `h` for short then displays the commands available for selection. `h command` provides a brief help text for the individual commands. `print` displays the partition table—in the following example, for a 1.5 TiB hard disk that already contains three partitions:

```
root# parted /dev/sda
(parted) print
Model: ATA WDC WD15EARS-00Z
Disk /dev/sda:  1500GB
```

```
Sector size (logical/physical): 512B/512B
Partition table: msdos

Number  Start   End     Size    Type     File system     Flags
 1      1049kB  50.0GB  50.0GB  primary  ext4            boot
 2      50.0GB  52.0GB  2000MB  primary  linux-swap(v1)
 3      52.0GB  84.2GB  32.2GB  primary
```

MB versus MiB, GB versus GiB

By default, `parted` displays partition limits and sizes in decimal units—that is, in `MB` = 10^6 bytes or `GB` = 10^9 bytes. Entries without a unit are interpreted as decimal MB. You can also explicitly specify the unit you want—for example, `100MiB` (binary MB) or `15GB` (decimal GB).

You can use `unit` to specify the desired unit for all inputs and outputs. Possible settings include `MiB` and `GiB` (binary counting, i.e., `GiB` = 2^{30} bytes) as well as `%` for percentages relative to the size of the hard disk:

```
(parted) unit MiB
(parted) print
...

Disk  /dev/sda:  1397GiB
...
Number     Start       End      Size  Type     File system     Flags
 1       1.00MiB  47684MiB  47683MiB  primary  ext4            boot
 2      47684MiB  49591MiB   1907MiB  primary  linux-swap(v1)
 3      49591MiB  80311MiB  30720MiB  primary
```

mklabel

If the hard disk is still completely unused and doesn't contain a partition table, you must set it up. To do this, you need to run `mklabel msdos` for MBR partitions or `mklabel gpt` for GPT partitions. Note that using `mklabel` means that you will lose all previously saved data on the hard disk!

mkpart and rm

You can create or delete partitions using the `mkpart` or `rm` commands. If you use `mkpart` for MBR partitions, you must specify the type (`primary`, `extended`, or `logical`) and the start and end position you require. For GPT partitions, the first parameter is interpreted as a freely selectable partition name:

```
(parted) mkpart primary 1MiB 10GiB     (for MBR)
(parted) mkpart part1   1MiB 10GiB     (for GPT)
```

> **Working without the Undo Function**
>
> In contrast to `fdisk`, where all changes to be made are initially only noted and only actually executed via the write command, `parted` executes every partitioning command immediately!

> Therefore, you should always make sure that you are editing the correct hard disk. If you do not specify a device when starting `parted`, `parted` automatically uses `/dev/sda`. This may not be intentional.

If you want to create a second partition after the first one, you should execute one of the following commands:

```
(parted) mkpart primary 10GiB 15GiB     (for MBR)
(parted) mkpart part2   10GiB 15GiB     (for GPT)
```

Older versions of `parted` complain at this point that the partitions would overlap. The easiest way out is to use `unit MiB` and `print` to determine the end of the previous partition and start the second partition 1 MiB further back. Given the size of today's data media, the 1 MiB gap created in this way is negligible.

Fortunately, this error message no longer occurs with current `parted` versions. The program is now intelligent enough to simply create the next partition after the previous one.

Partition types

Usually, `mkpart` creates partitions that later accommodate a Linux file system. This partition type is also suitable for software RAID and LVM. However, if you want to create a swap or Windows partition, you must specify the file system type in an additional parameter before the start position—for example, `linux-swap`, `fat32`, or `ntfs`. Further possible file system types are provided via `help mkpart`:

```
(parted) mkpart primary linux-swap 1MiB 10GiB     (MBR)
(parted) mkpart part1   linux-swap 1MiB 10GiB     (GPT)
```

Before you can create a logical partition on an MBR disk, you must create an extended partition (`mkpart extended start end`).

Enlarging partitions

You can use `resizepart <nr> <end>` to enlarge an existing partition, provided there is still space on the data medium behind it. If the partition already contains a file system, this must also be enlarged afterward. In reality, however, changing the partition size is tricky. `parted` makes the whole thing even worse with misleading error messages—even if it would be safe to change the size. You should therefore try to size the partitions correctly from the outset!

Deleting partitions

To delete a partition, you must first determine the numbers of all partitions via `print`. Then use `rm N` to delete the relevant partition, where `N` is a partition number according to `print`. When you delete a partition, no data is lost for the time being. For this reason, it's basically possible to undo this step

by recreating the partition with exactly the same start and end points. The problem, however, is that the position information must be absolutely right.

set

If you want to use the new partition as part of an LVM or RAID system, you must set additional attributes (flags) accordingly. The required command is `set partition number attribute name on/off`. Possible attributes include `boot`, `lvm`, `raid`, and `bios_grub`:

```
(parted) set 3 lvm on
```

EFI partition

To use `parted` to set up an EFI partition on a GPT volume, you must use `fat32` as the partition type and the `boot` and `esp` flags:

```
(parted) mkpart part1 fat32 1mib 250mib
(parted) set 1 boot on
(parted) set 1 esp on
```

Restoring deleted partitions

If you've deleted a partition by mistake, you can try to restore this partition using `rescue`. For this purpose, you must specify the approximate start and end points of the partition. `parted` then searches the disk for known file systems. This search succeeds faster depending on how exact the start and end points are. If `parted` finds it, it suggests inserting this partition back into the partition table:

```
(parted) rescue 200mib 400mib
searching for file systems...

A ext4 primary partition was found at 210MB -> 419MB.

Do you want to add it to the partition table?

Yes/No/Cancel? yes
```

You should not expect any miracles from the `rescue` command. It can only be successful if the file system is undamaged and has not already been partially overwritten by other, overlapping file systems, for example.

18.5.1 Example 1 (MBR)

The following commands first set up a primary partition (500 MiB), then an extended partition of the maximum size. A logical partition of 19.5 GiB is subsequently created in the extended partition. Note that the starting point for the logical partition is 1 MiB larger than the starting point for the

extended partition. This creates a small, unused gap, but the partitions are optimally aligned:

```
root# parted /dev/sdb
(parted) mklabel msdos
(parted) mkpart primary 1mib 500mib
(parted) mkpart extended 500mib 100%
(parted) mkpart logical 501mib 20gib
(parted) unit MiB
(parted) print
Number  Start    End       Size      Type      File system  Flags
 1      1.00MiB  500MiB    499MiB    primary
 2      500MiB   30720MiB  30220MiB  extended               lba
 5      501MiB   20480MiB  19979MiB  logical
```

Now another logical swap partition is to be created on the hard disk. The partition should be one GiB in size. However, version 3.5 (used for this example) fears an overlap here. That is why I have also built in a MiB gap. The starting point was the `print` result shown previously with `unit MiB`:

```
(parted) mkpart logical linux-swap 20481mib 21GiB
(parted) print
Number  Start     End       Size      Type      File system     Flags
 1      1.00MiB   500MiB    499MiB    primary
 2      500MiB    30720MiB  30220MiB  extended                  lba
 5      501MiB    20480MiB  19979MiB  logical
 6      20481MiB  21504MiB  1023MiB   logical   linux-swap(v1)  swap
```

18.5.2 Example 2 (GPT)

In a second example, two partitions are to be created on an SSD with GPT, one of which is to serve as a physical volume for an LVM system:

```
root# parted /dev/nvme0n1
(parted) mklabel gpt
(parted) mkpart part1    1mib  2000mib
(parted) mkpart part2 2001mib 64000mib
(parted) set 2 lvm on
(parted) print
Number  Start    End      Size       Name      File system     Flags
 1      0.00GiB  2.00GiB  2.00GiB    part1
 2      2.00GiB  64.0GiB  62.0GiB    part2                     lvm
```

18.6 Partitioning Tools with a Graphical User Interface

`parted` is admittedly not a command that offers much convenience. For this reason, I present alternative programs in this section, all of which have a graphical user interface. However, these programs are not a complete replacement for `parted`—because if manual partitioning is required, there is often no graphics system available!

GParted (MBR, GPT)

GParted is a graphical user interface for `parted` (see Figure 18.1). GParted can only change partitions that are currently not in use—that is, that are not included in the directory tree.

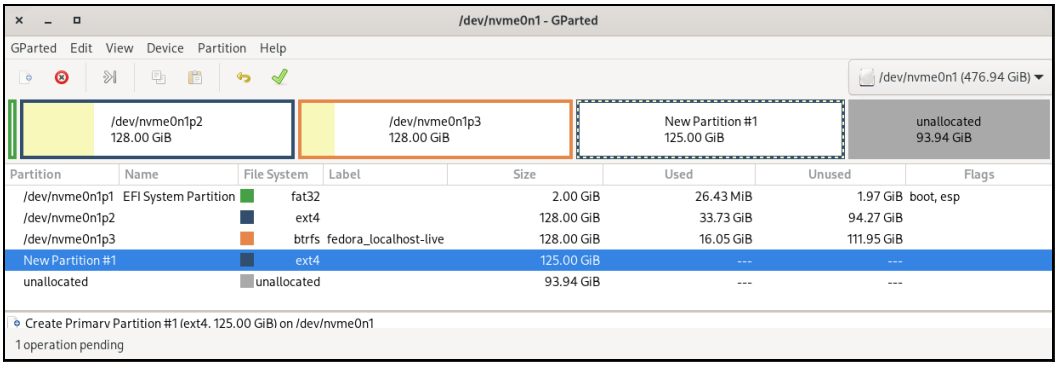


Figure 18.1 GParted

All used partitions are marked with a key in the program; the edit buttons are locked for these partitions. To change this, remove the relevant file systems from the directory tree (`umount`) or start GParted from a live image on a USB flash drive. Unlike `parted`, GParted remembers all actions, but does not execute them for the time being. **Edit • Undo** cancels actions, while **Edit • Apply** executes them.

To flag a partition for use for RAID or LVM, right-click the partition and execute **Manage Flags**. You can then set various flags that indicate the function of the partition.

gnome-disks (MBR and GPT)

The GNOME *Disks* program (`gnome-disks` command; see Figure 18.2) supports you in partitioning hard disks and SSDs, setting up file systems, and including file systems in the directory tree (including modifying `/etc/fstab`). In some distributions, the program is included in the `gnome-disk-utils` package and must be installed separately.

YaST

Under SUSE, you can also use the **System • Partition** YaST module as an alternative to GParted or `gnome-disks`.

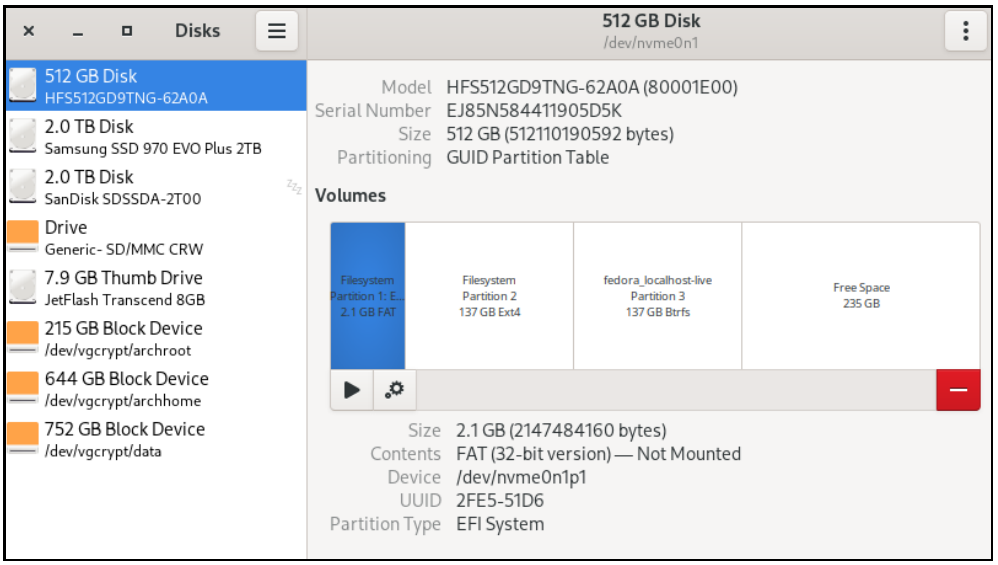


Figure 18.2 GNOME Disks Program

18.7 File System Types

This section provides an overview of the file system types that can be used on Linux. Some particularly important file system types will be discussed in more detail later in this chapter: `ext2` to `ext4`, `btrfs`, `xfs`, `vfat`, `ntfs`, and `iso9660`. You can easily determine which file system type(s) you are currently using via the `df -T` command.

Linux file systems

Linux file systems are suitable for the installation and operation of Linux. In everyday life, you won't even notice which of the file system types listed ahead that you're using. Elementary commands such as `ls` or `cp`, the administration of access rights, and so on: all this works independently of the file system.

Which Is the Best Linux File System?

There is no such thing as the "best" or "fastest" file system: every rating depends on the intended use. My recommendation is `ext4` for both desktop and server installations. `ext4` is a comparatively simple but very robust file system.

If you're interested in benchmark tests, you should visit *https://phoronix.com*. Michael Larabel publishes comparisons of the different file systems there from time to time.

ext4, btrfs, and xfs differ in features that are mainly relevant for advanced users or for server use—speed when dealing with very large files or with very many rather small files, efficiency in write and read operations, CPU load, journaling function (behavior after a crash), quota function (the ability to limit the maximum memory usage per user), management overhead, snapshot functions, encryption and compression functions, TRIM support for SSDs, and so on:

- **ext**
  ext2 (extended filesystem, version 2) was the dominant Linux file system in the early days of Linux. It was followed by ext3 in 2002, and ext4 has been the standard since 2008. Since then, the maximum file system size has been one exabyte (1,048,576 TiB), which should be enough for a while.
- **btrfs**
  This file system developed from scratch with the support of Oracle includes device mapper, snapshot, and RAID functions. Unfortunately, btrfs is very complex. Fedora and SUSE use btrfs by default. In RHEL, however, btrfs has been considered *deprecated* since version 7.4, which seems a bit harsh.
- **xfs**
  xfs was originally used as a file system on SGI workstations running the IRIX operating system. xfs is used by default on RHEL and its clones. Compared to ext4, xfs is considered to be particularly mature for file systems with more than 16 TiB of storage space—something that is only relevant for very large server installations.

ZFS Surprisingly, ZFS is regarded as the benchmark against which all file systems must be measured. ZFS was developed two decades ago by Sun for the Solaris UNIX operating system. Today, the rights to ZFS are owned by Oracle. As the ZFS code is not GPL-compatible, it cannot be integrated into the official Linux kernel code. However, it is possible to install the driver as a binary module or as source code (see *https://zfsonlinux.org*).

Windows, macOS The following file systems support data exchange with DOS, Windows, and macOS systems:

- **vfat**
  This file system, actually a combination of fat32 and vfat, was originally used as a file system for Windows 9x/ME. Today, it's still often used for small SD cards and USB flash drives (up to 32 GiB). Linux can read and write such file systems.
- **exfat**
  This further development of the vfat file system is the actual standard

for USB flash drives and SD cards with more than 32 GB of storage space. Since kernel version 5.7, Linux also supports this file system extremely well.

- **ntfs and ReFS**
  `ntfs` is used in all current Windows versions from Windows NT on. Linux copes well with `ntfs` and can read and write files.

  The newer *Resilient File System* (ReFS) can only be found on some Windows server installations. Rumor has it that it will be used more frequently in Windows 12. Linux support is poor; there is only a proprietary driver from Paragon.
- **hfs and hfsplus**
  These file systems are used on older Apple computers. Linux can read and write such file systems. However, writing only works if the file systems were set up on macOS *without* journaling functions.
- **apfs**
  The new *Apple File System* (`apfs`) has been in use on iOS and macOS since mid-2017. There is currently only a commercial `apfs` driver from Paragon for Linux.

CD/DVD

Data CDs and DVDs usually use their own file systems:

- **iso9660**
  The file system for CD-ROMs is defined by the ISO-9660 standard. However, this standard only provides for short file names. Long file names are supported by different extensions depending on the operating system (Rockridge, Joliet).
- **udf**
  The *Universal Disk Format* has established itself as the successor to ISO 9660. It is often used for DVDs.

Network file systems

File systems do not have to be located on local data media; they can also be integrated via a network. The Linux kernel supports various network file systems. I will only mention four representatives here:

- **nfs**
  The *Network File System* (NFS) is the classic network file system for Unix/Linux.
- **cifs**
  This file system, which used to be called `smbfs`, allows Windows or Samba network directories to be included in the directory tree.
- **sshfs**
  The rather rarely used `sshfs` file system makes it possible to integrate directories accessible via SSH into the local directory tree.

- **coda**
  This file system is most comparable to NFS. It provides numerous additional functions but is not very widespread.

Virtual file systems

Linux has a number of file systems that are not intended for storing data on a hard disk or other data media, but only for exchanging information between the kernel and application programs. In `/proc/filesystems`, these file systems are labeled `nodev`. Only the most important of these file systems are briefly described ahead:

- **devpts**
  This file system enables access to pseudo-terminals (PTYs for short) in accordance with the Unix 98 specification via `/dev/pts/*`. Pseudo-terminals emulate a serial interface and are used in terminal windows, for example.
- **proc and sysfs**
  The `proc` file system is used to map admin information of the kernel or process management. In addition, the `sysfs` file system maps the relationships between the kernel and the hardware. The two file systems are mounted at the `/proc` and `/sys` positions.
- **tmpfs**
  This temporary file system enables an efficient exchange of data between programs. Various `/run/xxx` directories are implemented with it. Data stored in these directories is lost when the computer restarts.
- **devtmpfs**
  The `devtmpfs` file system maps the device files in the `/dev` directory.
- **cgroup**
  Control groups make it possible to control or limit the use of resources by individual processes. The virtual *Cgroup Filesystem* helps to read or change the current cgroup settings.

Other file systems

Finally, here are some file systems or keywords that cannot be categorized into the preceding groups:

- **auto**
  There is no `auto` file system. However, `auto` may be used in `/etc/fstab` or when using `mount` to specify the file system. Linux then tries to recognize the file system itself. This works for most major file systems.
- **autofs**
  `autofs` is also not a separate file system, but a kernel extension that automatically executes `mount` for the file systems currently required. If the file system is no longer used for a while, `umount` is also executed automatically. This method is particularly useful if only a few of the numerous

NFS directories are actively used. A more modern alternative to `autofs` is provided by systemd (see `man systemd.mount`).

- **cramfs and squashfs**
  The *cram file system* and the *squash file system* are read-only file systems. They are used to pack as much data as possible in compressed form into flash memory or read-only memory (ROM). The squash file system is used by the Snap package management system (Ubuntu) to integrate Snap files into the directory tree as read-only file systems.
- **fuse**
  Filesystem in Userspace (FUSE) makes it possible to develop and use file system drivers outside the kernel. FUSE is therefore always used together with external file system drivers.
- **gfs and ocfs**
  The *Global File System* (GFS) and the *Oracle Cluster File System* (OCFS) enable the creation of huge, networked file systems that can be accessed by multiple computers at the same time.
- **Loop**
  The *loopback device* is an adapter that can address an ordinary file like a block device. This allows you to place any file system in a normal file and integrate it into the directory tree using `mount`. The corresponding *Loopback Device Support* kernel function is implemented in the `loop` module. The loopback device is used, for example, to create an initial RAM disk for GRUB, to use Snap packages (Ubuntu), or to test ISO images.
- **none**
  Of course, `none` is not a file system either. However, it's possible to include a local directory at a different location in the directory tree. To do this, you must specify `none` as the file system type for `mount` or in `/etc/fstab` and `bind` as an additional option. The effect is similar to a symbolic link, but the internal implementation is completely different. This procedure is useful, for example, when you configure an NFS4 server.
- **unionfs/aufs/mhddfs/overlay**
  The concept of `unionfs` or its variant `aufs` makes it possible to virtually superimpose multiple file systems, whereby the topmost file system has priority. `unionfs` and `aufs` are used in some live systems: Linux boots directly from the DVD or a USB flash drive. A RAM disk file system is superimposed on the file system of the data medium, in which changes can be made.

  Currently, the most modern implementation of an overlay file system is `overlay2`. This file system is used by Docker, for example, to save changes to a base image.

- **Encrypted file systems**
  Linux has various methods for encrypting the content of file systems. Some of these methods are based directly on their own file systems (e.g., CryptoFS or eCryptfs). However, the combination of LVM and encryption is more common.

You can see which file systems are directly integrated into the running kernel or are currently loaded as a module in the `/proc/filesystems` file.

18.8 mount, umount, and /etc/fstab

After installing Linux, you usually do not have to worry about managing the file system: you can access all or at least most of the data partitions on the hard disk via various directories.

This section takes a look behind the scenes and describes the `mount` and `umount` commands as well as the `/etc/fstab` file:

- `mount` or `umount` is always executed when a file system is integrated into the directory tree or removed from it again. Of course, you can also run these commands yourself as `root` if the automatisms fail or if you do not use a graphical desktop system.
- The `/etc/fstab` configuration file controls which file systems are automatically integrated into the directory tree when the computer is started and which options apply. `/etc/fstab` is preconfigured during the Linux installation. If you are not happy with this configuration or if your requirements change later, you must change the file using an editor. This section describes the syntax of this file.

Surprisingly, there are only a few graphical configuration tools that support the `mount` process or the modification of `/etc/fstab`. Exceptions include the `gnome-disks` program briefly described in the previous section and the **System • Partition** YaST module (on SUSE).

External hard disks or USB flash drives represent a special case: most distributions automatically integrate these types of data media into the file system as soon as they are connected to the computer.

18.8.1 Determining the Current State of the File System

df

If you want to know how your Linux system is currently organized, the easiest way is to execute the `df -h` command. This command shows where hard disks, data media, and so on are mounted in the file system and how much space is still available on the individual hard disks. (The `duf` command,

which must be installed separately, provides more visually appealing results.)

mount

The `mount` command without any additional options provides even more detailed information about the mounted file systems. The command also shows all active `mount` options. Unfortunately, the really interesting entries are often lost among the countless virtual file systems.

In the following example, the output has been indented in columns to improve readability. You can achieve this clear display by processing the result of `mount` using `column`:

```
user$ mount | column -t
/dev/nvme0n1p1   on /boot/efi            type vfat     (rw,relatime,fmask=...)
/dev/mapper/p... on /crypt               type ext4     (rw,noatime,errors=...)
/dev/sda1        on /media/kofler/p...   type ext4     (rw,nosuid,nodev,...)
binfmt_misc      on /proc/sys/fs/bi...   type bin...   (rw,relatime)
gvfsd-fuse       on /run/user/1000/gvfs  type fuse...  (rw,nosuid,nodev,...)
cgroup           on /sys/fs/cgroup/...   type cgroup   (rw,nosuid,nodev,...)
cgroup           on /sys/fs/cgroup/...   type cgroup   (rw,nosuid,nodev,...)
devpts           on /dev/pts             type devpts   (rw,nosuid,noexec,...)
sysfs            on /sys                 type sysfs    (rw,nosuid,nodev,...)
proc             on /proc                type proc     (rw,nosuid,nodev,...)
udev             on /dev                 type devtmpfs (rw,nosuid,...)
tmpfs            on /run/user/1000       type tmpfs    (rw,nosuid,nodev,...)
...
```

`findmnt` provides the same results as `mount`. The advantage of this command is the much more reader-friendly formatting of the output. The `/etc/mtab` and `/proc/mounts` files also provide similar information to `mount`. A basic problem with `mount`, `findmnt`, and similar commands is that they often provide way too much information. You're usually not interested in the endless list of temporary file systems and management data. If you only want to see "real" file systems, you should use the `-t` option to specify the file system types relevant to you:

```
user$ fndmnt -t ext4,btrfs,xfs,vfat
```

18.8.2 Mounting and Unmounting File Systems Manually (mount and umount)

After installing a current Linux distribution, the system is configured in such a way that you will rarely need `mount`: all Linux file systems are mounted in the directory tree. A new window of the KDE or GNOME file manager appears automatically when external data media are connected. Even if it may seem as if the whole thing works like magic, the `mount`

command is always executed behind the scenes to integrate file systems into the directory tree or remove them from it.

The syntax of `mount` is as follows:

```
mount [options] device directory
```

Among other things, the file system type is specified in the options (`-t xxx`). The device name designates the partition or drive. Any directory of the current file system can be specified as the directory. This directory must already exist. If necessary, you can create it using `mkdir`! In general, `mount` can only be executed by `root`. However, it's possible for `/etc/fstab` to allow all users to execute `mount` for individual partitions (`user` or `users` option).

Example

The easiest way to understand `mount` is to look at a few examples. The first example enables access to the data of a Windows partition via the `/windows` directory:

```
root# mkdir /windows
root# mount -t ntfs /dev/sda2 /windows
```

Be Careful when Accessing Active Windows File Systems!

This example assumes that the Windows file system is not in use—that is, that Windows has been completely shut down. On a computer with a parallel installation of Linux and Windows, this requirement is often not met! By default, Windows is only put into a kind of hibernation state when it is switched off, which allows for a quick restart. A change to the Windows file system by Linux can then lead to a loss of data.

To address this problem, shut down Windows completely by using `shutdown /p` or use a cloud service such as NextCloud to synchronize files that you want to edit in both operating systems.

remount

You can use `mount -o remount` to change the options of an already mounted file system. For example, the following command activates the `exec` option so that programs contained therein can be executed:

```
root# mount /media/my-disk -o remount,exec
```

If problems occur when mounting the system partition during the computer startup, the partition is only mounted read-only. However, to fix the cause of the error, such as an incorrect entry in `/etc/fstab`, it's often necessary to make changes to the file system. To do this, you can run the following command. It remounts the system partition, whereby write access is then also possible:

```
root# mount -o remount,rw /
```

To remove a file system from the directory tree, you need to execute `umount`: **umount**

```
root# umount /media/dvd
```

18.8.3 Mounting File Systems Automatically (/etc/fstab)

It would be very tedious if you had to remount various partitions after every system start. The key to making your work easier is `/etc/fstab` (*fstab* here meaning *file system table*). This file specifies which data media are included in the file system at system startup. In any case, `fstab` must contain the system partition and all file systems required for internal administration.

Depending on the distribution, a minimal `fstab` file may look as follows. The second partition of the first hard disk/SSD contains the root file system. There are often additional entries in `fstab` for the `/home`, `/boot`, or `/boot/efi` directories as well as for a swap partition or swap file: **Examples**

```
# file /etc/fstab
/dev/sda2                   /      ext4  defaults             0 0
```

On my primary notebook computer running Arch Linux, the setup is a bit more complicated. `/boot` is the path to the EFI partition. The remaining three file systems are located in logical volumes in an encrypted partition. (Details on LVM and encryption follow later in Section 18.16 and Section 18.19.) `/etc/fstab` has the following content:

```
UUID=7a7872cd-5c73-4271-...   /      ext4  rw,relatime          0 1
UUID=E4E5-DA46                /boot  vfat  rw,relatime,utf8,...  0 2
/dev/mapper/vgcrypt-archhome  /home  ext4  rw,relatime          0 2
/dev/mapper/vgcrypt-data      /data  ext4  rw,relatime          0 2
```

18.8.4 The Syntax in /etc/fstab

The preceding examples show the basic format of `fstab`: each line describes a data medium (a partition, a file system) in six columns.

The first column contains the device name of the data medium. Instead of the device name, you can also enter the *volume name* or the ID number of the file system. The correct syntax in this case is `LABEL=string` or `UUID=nnn-nnn`. You can use `blkid` to determine the partition name and the UUID of a partition: **First column**

```
root# blkid /dev/sda9
/dev/sda9: UUID="5a954fc1-00c6-4c25-a943-d4220eff350d" TYPE="ext4"
```

To change this data, you can use different tools depending on the file system—for example, `tune2fs`.

The advantage of labels or UUIDs compared to device names is that the information is still correct even if the device name has changed. This can easily happen with NVMe SSDs and USB data media in particular. Depending on the order in which the kernel recognizes the data media, the device name can change. Unfortunately, `fstab` gets very confusing when you use UUIDs.

Second column

The second column specifies the directory in which the data medium is integrated into the file tree. The directories specified in the second column must already exist. They do not have to be empty, but once the file system has been mounted, you will no longer be able to access the files they contain, only the files on the mounted data medium.

Third column

The third column specifies the file system. Table 18.4 lists the most important file systems in alphabetical order. You can also specify multiple file systems separated by commas. `mount` automatically chooses the correct one of the available systems. The file system names must not be separated by spaces.

File System	Usage
`auto`	Automatically detecting a file system
`btrfs`	`btrfs` file system
`cifs`	Windows network directory (Samba)
`ext2`, `-3`, `-4`	`ext` file system versions 2, 3, and 4
`iso9660`	Data CDs
`nfs`	Unix network directory (NFS)
`ntfs`	Windows file system
`proc`	Process management (`/proc`)
`smbfs`	Windows network directory (Samba)
`swap`	Swap partitions or files
`sysfs`	System administration (`/sys`)
`tmpfs`	Temporary file system
`udf`	Universal disk format (DVDs, CD-RWs)

Table 18.4 File Systems

File System	Usage
vfat	Windows 9x/ME file system
xfs	XFS file system

Table 18.4 File Systems (Cont.)

Fourth column

The fourth column determines options for accessing the data medium.

Multiple options are separated by commas. Again, no spaces may be inserted! Table 18.5 lists the most important universal `mount` options. If you don't want to use any option at all, you need to enter `defaults` (note the plural!).

Option	Meaning
defaults	Using default options.
dev	Analyzing the marking of character or block devices.
discard	Activating SSD trim (ext4, btrfs, xfs, and swap).
exec	Allowing the program execution (e.g., for CD/DVD drives).
noauto	Not mounting the disk at system startup.
nodev	Ignoring the marking of character or block devices.
noexec	No program execution allowed.
nofail	Continuing the boot process if the file system does not exist.
nosuid	Not analyzing suid and guid access bits.
owner	The owner may execute (u)mount.
ro	Read-only (write protection).
sw	Swap (swap file or partition).
suid	Analyzing suid and guid access bits.
sync	No buffering of write accesses (safer, but slower).
user	Anyone may run mount, but only the user of the last mount call may run umount.
users	Anyone may perform mount and umount.

Table 18.5 mount Options

Fifth column The fifth column contains information for the `dump` program and is ignored by Linux. It is common to enter 1 for the system partition and 0 for all other partitions or disks.

Sixth column The sixth column specifies whether and in what order the file systems should be checked at system startup. Often 1 is entered for the system partition and 0 for all other partitions. This means that only the system partition is checked for errors and repaired if necessary when the computer is started.

If you want other partitions to be checked automatically, you should enter the number 2 for these partitions; that is, the check should take place after the system partition has been checked. If entries in the fifth and sixth columns in `/etc/fstab` are missing, 0 is assumed.

With `ext` file systems, you can also use `tune2fs` to influence how often an automatic check should be carried out, such as once a year or after 50 `mount` operations.

Avoid Syntax Errors!

Many distributions hang at system startup if you make a syntax error in `/etc/fstab`! This is not quite so bad with a computer in front of you; you just need to log in to a text console and correct the error in an editor.

A syntax error like this is far more unpleasant on a server that is located somewhere far away. The server no longer boots up properly, and in the absence of a network connection, you have no chance of fixing the error! You often don't even know why the server has suddenly stopped starting. (If the server is running in a hosting company, then the company often offers the option of starting a network image with a rescue system to correct the problem.)

To quickly test whether a new entry in `/etc/fstab` is correct, you should run `umount /mountdir` and `mount /mountdir`, specifying the directory from the second `fstab` line instead of `mountdir`. `mount` then reads the remaining data from `/etc/fstab` and immediately corrects any errors.

systemd Changes to `/etc/fstab` only take effect when you then execute `mount` for the affected file system. Depending on the configuration, systemd (see Chapter 20) also analyzes `/etc/fstab`. For this to work, you must explicitly inform systemd that the file has changed:

```
root# systemctl daemon-reload
```

18.8.5 Bind Mounts

Linux provides the option of using bind mounts to integrate an existing directory at a different location in the directory tree as a new file system. The content of this directory is then accessible via two locations in the directory tree.

The easiest way to understand this mechanism is by means of an example. Let's assume that you have set up a root file system with 100 GiB and a home file system with 1 TiB when installing Linux. Now it turns out that the root file system is too small because the image files of various virtual machines in `/var/lib/libvirt` take up a lot of space. You therefore want to save the virtual machines in the home file system.

To do this, you temporarily stop all virtual machines and move the `/var/lib/libvirt` directory tree to `/home/libvirt`:

```
root# mv /var/lib/libvirt /home
```

To ensure that the directory tree moved to another file system can still be used with its original path, you must set up the directory again and run `mount -o bind`:

```
root# mkdir /var/lib/libvirt
root# mount -o bind /home/libvirt /var/lib/libvirt
```

To automate this process, add the following line to `/etc/fstab`:

```
# file /etc/fstab
...
/home/libvirt /var/lib/libvirt bind bind 0 0
```

Other possible uses of bind mounts are summarized at *https://unix.stackexchange.com/questions/198590/what-is-a-bind-mount*.

Bind mounts versus symbolic links

Instead of the bind mount, a simple symbolic link would have sufficed:

```
root# umount /var/lib/libvirt
root# ln -s /home/libvirt /var/lib/libvirt
```

The main advantage of the bind mount is that it is well-documented by the entry in `/etc/fstab`. A third-party administrator would find the symbolic link difficult to discover.

On the other hand, many backup tools do not work well with bind mounts. There is a risk that the data in `/var/lib/libvirt` and in `/home/libvirt` will be backed up twice during a system backup. On the Stack Exchange website, most authors therefore recommend giving preference to symbolic links (see *https://unix.stackexchange.com/questions/49623*).

18.8.6 Automatic Mounts without /etc/fstab

In many Linux distributions, `/etc/fstab` consists of only a few lines for the root file system, the swap partition, and possibly a home file system. The entry for the root file system primarily fulfills a documentation function: the device for the root file system must be passed to the kernel during the boot process anyway. (Usually GRUB takes care of this.) This leaves one or two lines that specify the location of the swap partition or the file system for the home directory.

In such simple cases, the systemd init system can manage without `/etc/fstab`. However, the home and swap partitions must be identified with special partition type identification numbers (also called *partition type GUIDs* or *partition GUID code*)—for example, 933ac7e1-2eb4-4f13-b844-0e14e2aef915 for the home partition or 0657fd6d-a4ab-43c4-84e5-0933c84b4f4f for the swap partition. Further GUIDs are documented in the Discoverable Partitions Specification and on Wikipedia:

- *https://www.freedesktop.org/wiki/Specifications/DiscoverablePartitionsSpec*
- *https://en.wikipedia.org/wiki/GUID_Partition_Table#Partition_type_GUIDs*

Note that these numbers are neither file system UUIDs, which are randomly generated when setting up a file system, nor the partition UUIDs used to identify all partitions in GPT. Partition type identifiers are rather a third number that can be stored together with a GPT partition. You can determine these identification numbers using `blkid -s PART_ENTRY_TYPE` (the following output was created on Clear Linux; the partition names also given make it easier to interpret but are not part of the Discoverable Partitions Specification):

```
root# blkid -p -s PART_ENTRY_NAME -s PART_ENTRY_TYPE /dev/sda?
/dev/sda1: PART_ENTRY_NAME="EFI"
           PART_ENTRY_TYPE="c12a7328-f81f-11d2-ba4b-00a0c93ec93b"
/dev/sda2: PART_ENTRY_NAME="linux-swap"
           PART_ENTRY_TYPE="0657fd6d-a4ab-43c4-84e5-0933c84b4f4f"
/dev/sda3: PART_ENTRY_NAME="/"
           PART_ENTRY_TYPE="4f68bce3-e8cd-4db1-96e7-fbcaf984b709"
```

During the boot process, the `systemd-gpt-auto-generator` takes care of discovering partitions that comply with the Discoverable Partitions Specification and automatically includes them in the directory tree or uses them as swap space. If you want to set the partition type identification number of a partition yourself, you can use the `gdisk` partition editor (menu command T).

Clear Linux

There are currently only a few distributions that do not use `/etc/fstab` by default. Clear Linux is one of these few exceptions. In the default image of this distribution there are three partitions, one each for EFI, the root file system, and the swap memory. However, the EFI partition is not included in the directory tree by default. Optionally, `/etc/fstab` is of course also permitted on Clear Linux in order to include additional partitions for which the Discoverable Partitions Specification does not provide a rule. Details of the automounting process in Clear Linux can be found at *https://clearlinux.org/news-blogs/where-etcfstab-clear-linux*.

18.9 Basic Principles of File Systems

The following pages focus on the `ext2`, `ext3`, `ext4`, `btrfs`, and `xfs` Linux file systems. Before I describe their setup and administration, this section summarizes some basic information that applies regardless of the file system type.

Journaling

All common Linux file systems support journaling functions. In its simplest form, *journaling* means that the start and end of each file operation is logged in a special file or in a reserved area of the file system. Thanks to this log, you can check later whether a particular file operation has been fully executed. If it hasn't, the operation can be revoked. In the database world, this is referred to as a *transaction*.

With advanced journaling systems, it's possible to log the actual changes to the files in the journal. This slows down normal operation but provides more options for a later reconstruction.

If a file operation cannot be fully completed (power failure, kernel crash, etc.), this is shown in the log. With simple journaling, the changes are lost, but the previous state of the file is usually still available.

The great advantage of journaling functions is that the file system is very quickly restored to a consistent state the next time the computer is started and can be used again almost immediately. This is a big difference compared to the past, when the entire file system had to be systematically searched for any errors after a crash or power failure. This took several minutes, or even hours in the case of very large data media.

Data loss despite journaling

In the event of a power failure, even journaling does not guarantee a consistent file system! The problem can be found with hard disks or SSDs: for reasons of efficiency, they use an internal cache when writing. It can therefore happen that the file system receives confirmation from the data medium that it has received and saved the data. However, it can still take seconds for the data to be physically written from the cache to the hard disk. Although

this time span is much shorter with SSDs, it does not change the fundamental problem.

If a power failure occurs during this period, the data in the cache is lost. This cache can be deactivated on some hard disks. However, this reduces the speed of write operations to such an extent that it is usually not used in practice.

Automatic check of the file system

If Linux detects at startup that the computer was not shut down properly last time, it performs a file system check for the system partition and, depending on the configuration, for other partitions named in `/etc/fstab`. Whether a check takes place or not is decided by the sixth column in `/etc/fstab`. Thanks to the journaling features, this review is usually done in no time at all.

Apart from this, some file systems (including `ext` in all versions) provide for a regular check of the file system for consistency errors. These relatively time-consuming checks occur when the computer is started if a certain amount of time or number of `mount` operations has been exceeded since the last check. After the introduction of journaling functions, it was often argued that the regular consistency test was now superfluous. Unfortunately, this is not quite true: a file system can also become inconsistent due to hard disk hardware errors—and the probability of such errors increases with increasing hard disk size! For example, in the data sheet of a 1 TiB hard disk, I found the specification that the probability of bit errors (*nonrecoverable read errors per bits read*) is less than 1 in 10^{15}. That sounds really negligible. However, if you take into account that there is room for 8×10^{12} bits on this hard disk, it becomes clear that data errors in regular operation—that is, without any damage—are not particularly unlikely. While a regular consistency check of the file system cannot prevent these errors, it does provide some chance of detecting and correcting misbehavior, at least when areas critical to the internal management of the file system are affected.

Unfortunately, none of the file systems presented in this book are truly fault-tolerant. However, file systems that can detect hardware errors through checksums or even correct such errors through redundant storage are a highly topical area of research.

Manual check of the file system

You can easily perform a manual check using the `fsck` command. However, the partition in question must not be used during the check; that is, you must execute `umount` prior to the check. But you cannot check the system partition while the system is running because you cannot unmount the file system via `umount`. Instead, you need to run the `touch /forcefsck` command as `root` and restart the computer. The `forcefsck` file is also created when you run `shutdown` with the additional `-F` option.

If the /forcefsck file exists, almost all distributions automatically run a file system check at the next boot. If that doesn't work, boot the computer with a rescue or live system and run fsck from there.

Size limits

In the past, the question of the maximum size of files has come up again and again. The answer depends on which kernel, CPU architecture, glibc library, and file system you are using.

Current distributions consistently support the large file support (LFS) extensions in the glibc library. Thanks to LFS, the file size is almost unlimited with 2^{63} bytes. On the other hand, the various file system types also specify limits for the maximum file (system) size. Table 18.6 summarizes this data. In this context, the following applies: 1 TiB = 1024 GiB.

File System	Maximum File Size	Maximum File System Size
btrfs	16,777,216 TiB	16,777,216 TiB = 16 Exabytes = 16 EiB
ext3	2 TiB	32 TiB (at 8 KiB block size)
ext4	16 TiB	1,048,576 TiB = 1 EiB
xfs	9,437,184 TiB	9,437,184 TiB
ZFS	16,777,216 TiB	16,777,216 TiB

Table 18.6 Maximum File System Size

Changing the file system type

It's impossible to convert the file system type. The only option is to set up a new file system of the required type and then copy all the files.

18.10 The ext File System (ext2, ext3, and ext4)

Versions

The various ext versions dominate the world of Linux file systems. The first version was used only briefly in the initial phase of Linux. Then the ext2, ext3, and ext4 variants followed in 1993, 2002, and 2008. The maximum file system size grew from 2 GiB to one exabyte (1,048,576 TiB). Journaling functions have been available since ext3. The compatibility of the different ext file system versions is expressed by the fact that various administration tools still have the version number 2 in the command name, although they can also be used for newer versions: tune2fs or resize2fs thus also work with an ext4 file system.

The development of ext has by no means stood still since the release of ext4. While there are currently no plans for ext5, new features are being added to ext4 all the time.

/etc/fstab

Entries for ext4 file systems in /etc/fstab usually look like the following example:

```
# /etc/fstab: Linux file systems
/dev/sdb8   /          ext4   defaults      1 1
/dev/sdb9   /boot      ext4   defaults      0 0
/dev/sdb9   /data      ext4   relatime      0 0
```

18.10.1 Administration

Setting up the source system

The ext2, ext3, and ext4 file systems are formatted using mkfs.ext2, mkfs.ext3, or mkfs.ext4. In the following example, an ext4 file system is set up in a 30 GiB partition. mkfs.ext4 independently decides on a block size of 4 KiB and on 1,966,080 inodes:

```
root# mkfs.ext4 /dev/sdb1
Discarding device blocks: done
Creating filesystem with 7863808 4k blocks and 1966080 inodes
Filesystem UUID: 8d86fd1e-86a5-42e9-a400-ccf133ab0542
Superblock backups stored on blocks:
  32768, 98304, 163840, 229376, 294912, 819200,
  884736, 1605632, 2654208, 4096000
Allocating group tables: done
Writing inode tables: done
Creating journal (32768 blocks): done
Writing superblocks and filesystem accounting information: done
```

This means that you can create a maximum of just under two million files in the file system. The average file size would then be 16 KiB. If you want to store a larger number of smaller files or fewer larger files, you can use -i <n> to specify after how many bytes each inode should be provided. If the average file size is smaller than n, it isn't the size of the partition but the inode count that limits the file system.

Note that the absolute number of inodes can no longer be changed, even if the file system is later enlarged! In most cases, the default value of mkfs.ext4 is appropriate.

File system check

ext file systems can be periodically checked for errors at boot time, after a certain number of mount operations, or after a certain time, depending on which criterion was previously met. You can set the corresponding parameters using tune2fs. The -c option can be used to specify the maximum mount count, while -i specifies the time interval in days. You must also ensure that the sixth column of the corresponding fstab line contains a value greater than 0:

```
root# tune2fs /dev/sdb1 -c 30 -i 365
Setting maximal mount count to 30
Setting interval between checks to 31536000 seconds
```

To disable the function, you must specify the 0 values in each case or set the sixth column in /etf/fstab to 0. You can determine the current intervals for the automatic file system check by using tune2fs -l:

```
root# tune2fs /dev/sdb1 -l | egrep -i 'check|mount count'
Mount count:              5
Maximum mount count:      30
Last checked:             Jun 11 15:30:42 2023
Check interval:           31536000 (12 months, 5 days)
...
```

Despite journaling features, a file system check is recommended every now and then—and at least once a year. On the one hand, possible hardware errors of the hard disk are detected this way. On the other hand, the file system drivers may still contain unknown errors. The sooner the resulting errors are fixed, the smaller the potential damage.

You can perform a manual check at any time using the fsck.ext2/ext3/ext4 command. However, the partition in question must not be currently in use during the check; that is, you may have to perform umount beforehand:

```
root# fsck.ext4 -f /dev/sdb1
Pass 1: Checking inodes, blocks, and sizes
Pass 2: Checking directory structure
Pass 3: Checking directory connectivity
Pass 4: Checking reference counts
Pass 5: Checking group summary information
/dev/sdb1: 2907/1966080 files (0.1% non-contiguous), 175406/
7863808 blocks
```

Most of the time, the check reveals that everything is fine.

Otherwise, the remains of files that can no longer be reconstructed are stored in the /lost+found directory of the respective partition. If these were text files, you may still be able to extract useful information from the remains.

Setting the partition name

Using e2label, you can determine or set the internal name of an ext file system (the file system volume name). You can specify this name in the first column of /etc/fstab instead of the device name:

```
root# e2label /dev/sdb1 mylabel
```

Setting the UUID

During setup, the file system is automatically assigned a UUID, which you can determine via `blkid`. If necessary, you can change this number using `tune2fs -U`. The change can be made during operation; `umount` is not required:

```
root# tune2fs -U random /dev/
sdb1                                    (random UUID)
root# tune2fs -U f7c49568-8955-4ffa-9f52-9b2ba9877021 /dev/
sdb1 (custom UUID)
```

Changing the size of the file system

Using `resize2fs`, you can resize an `ext` file system. Note that if you enlarge the file system, you must *first* enlarge the underlying partition, logical volume (LV), or virtual disk—but if you want to shrink the file system, you must not shrink the partition, LV, or virtual disk until *afterward*! In the following example, the LV is enlarged using `lvextend`. More details about LVM administration follow in Section 18.16:

```
root# lvextend -L 40G /dev/mapper/vg1-test
  Extending logical volume test to 40,00 GB
  Logical volume test successfully resized
root# resize2fs /dev/mapper/vg1-test
Filesystem at /dev/mapper/vg1-test is mounted on /mnt/tst;
on-line resizing required
old_desc_blocks = 4, new_desc_blocks = 5
The filesystem on /dev/mapper/vg1-
test is now 10485760 (4k) blocks long.
```

Increasing the size of the file system is possible on the fly, as in the preceding example. However, an increase above 16 TiB is problematic according to various reports on the internet. I have not been able to test the process myself due to the lack of a sufficiently large RAID array. For downsizing, the file system must first be removed from the directory tree (`umount`).

Windows access to ext4 file systems

You can also access Linux partitions on Windows, for example using WSL2: *https://learn.microsoft.com/en-us/windows/wsl/wsl2-mount-disk*.

18.11 The btrfs File System

`btrfs` is currently the most modern file system for Linux. It combines many features that other file systems like `ext4` or `xfs` can only provide in combination with external components (RAID, LVM). The following list summarizes the most important features of `btrfs`:

- With copy-on-write (COW), modified file blocks are not overwritten but stored in a different location. In combination with journaling, this enables particularly secure file changes.
- Automatic calculation of checksums can detect bit errors.
- Direct support is available for RAID-0, -1, -5, -6 and -10.
- Snapshots and subvolumes are available.
- Compression of the files is possible (`mount` option `compress`).
- SSD optimization is an option.
- Defragmentation occurs during operation.
- Deduplication allows you to store files with the same content only once.
- You can send/receive messages about changes in the file system, enabling efficient synchronization and backup tools.

What speaks against `btrfs` is its high complexity, which overwhelms many Linux users. Even seemingly trivial questions, such as how much space is left on a file system, are difficult to answer in `btrfs`.

The development of `btrfs` was originally initiated by Oracle but is now done in cooperation with numerous other companies and kernel developers. The `btrfs` driver is integrated into the kernel and, like all kernel code, is subject to the GPL license.

Is btrfs the file system of the future?

For many years, `btrfs` was said to be the Linux file system of the future. By now, you might think the future is here: Fedora, openSUSE, and SUSE Enterprise products use `btrfs` as the default file system for the system partition. Various NAS devices from Synology also use `btrfs` (usually without their users realizing it). Facebook even has `btrfs` in use on millions of servers (see *https://code.fb.com/open-source/linux*).

The big downer, however, is that Red Hat, of all companies, has turned away from `btrfs`. For RHEL 7, `btrfs` has been considered deprecated since 2017, and in RHEL 8 the required drivers were removed altogether. This makes it impossible to set up a `btrfs` file system during installation. It is unclear whether there are technical reasons behind this decision or rather corporate policy considerations (the `btrfs` development is strongly influenced by Red Hat's competitor Oracle).

Canonical is more neutral toward `btrfs`. The default file system for Ubuntu is `ext4`. Although `xfs` and `btrfs` are also supported, these file systems are not specifically advertised and are accordingly rarely used.

Long story short, even though `btrfs` is considered mature today, it's questionable whether `btrfs` will ever replace `ext4` as the Linux default file system.

Links

There is comprehensive information about `btrfs` available on the internet. Before using `btrfs` for the first time, I recommend you take a look at the following page: *https://btrfs.readthedocs.io/en/latest*.

18.11.1 Administration

`btrfs` administration is mainly done by using the `btrfs` command. This command can be found together with `mkfs.brtfs` in the `btrfs-tools`, `btrfsprogs`, or `btrfsprogs` package depending on the distribution. If you haven't already set up a `btrfs` file system during the installation, you usually need to install this package separately.

Setting up a btrfs file system

To set up a new `btrfs` file system in an empty partition or logical volume, you'll want to run the following command (replacing `/dev/sdb1` with your own device name, of course):

```
root# mkfs.btrfs /dev/sdb1
Filesystem size:    40.00GiB
Block group profiles:
  Data:             single            8.00MiB
  Metadata:         DUP             256.00MiB
  System:           DUP               8.00MiB
SSD detected:       no
Zoned device:       no
Incompat features:  extref, skinny-metadata, no-holes
Runtime features:   free-space-tree
Checksum:           crc32c
Number of devices:  1
Devices:
   ID        SIZE  PATH
    1    40.00GiB  /dev/mapper/sdb1
```

Then you mount the file system in the directory tree:

```
root# mkdir /mnt/btrfs
root# mount /dev/sdb1 /mnt/btrfs
```

Enlarging/reducing the file system

If a `btrfs` file system turns out to be too small, the easiest solution is to add another device (i.e., a disk partition or a logical device). Details follow in Section 18.11.6.

However, it's also possible to increase and even decrease the size of an existing `btrfs` file system on the fly. In practice, this works best when the file system is in a logical volume, although this is uncommon with `btrfs`.

To enlarge a btrfs file system so that it completely uses a logical volume that was previously enlarged by using lvextend, you need to run the following command:

```
root# btrfs filesystem resize max /mnt-point
```

Instead of max, you can also specify the new absolute size of the file system or use + or - to indicate the relative change. The abbreviations k, m, and g for KiB, MiB, and GiB are allowed. The following command shrinks the file system by 2 GiB:

```
root# btrfs filesystem resize -2g /mnt-point
```

Checking the file system

To check the consistency of the file system, you can use the btrfs check command, which can be run only for unmounted file systems. The only parameter you pass is the device name of the partition with the btrfs file system:

```
root# btrfs check /dev/sdb1
```

btrfs check analyzes the file system but does not make any changes. If you want to try to fix errors in the file system, you must also specify the --repair option.

18.11.2 Deactivating Copy-On-Write

btrfs is based on the COW method. When a file is modified, these changes are stored in empty blocks of the file system. The original blocks will be released later.

This is a reasonable approach, but in exceptional cases COW leads to unnecessary file fragmentation and decreased performance, which particularly affects images of virtualization programs as well as binary database files—that is, large files in which changes are made block by block.

I recommend that you avoid btrfs altogether for database and virtualization files and rather use a separate ext4 or xfs file system for the directory in question. But if you want to store databases or images inside btrfs, then you can set the C attribute for these files or even for entire directories by using chattr:

```
root# touch file
root# chattr +C file
root# chattr +C directory
```

chattr +C deactivates COW for a file or an entire directory. In the second case, however, the attribute applies only to new files created in that directory. To disable COW on existing files with a size not equal to 0 bytes, you must copy the files:

```
root# mv dir backup
root# mkdir dir
root# chattr +C dir
root# cp -a backup/* dir
root# rm -rf backup
```

18.11.3 Compressing Files

btrfs supports the automatic and transparent compression of files. For this purpose, the file system must be mounted in the directory tree with the mount option compress=zlib, compress=lzo, or compress=zstd. The compress option applies only to new or modified files. Existing files remain unchanged as long as they are read-only. The option applies to the entire file system, so it cannot be enabled only for individual directories.

The three compression methods differ in their efficiency and speed: lzo is the fastest algorithm, but zlib and zstd save more space.

compress can accelerate file operations on conventional hard disks. At first glance, this may seem surprising as compression or decompression causes additional effort during reading. With a fast CPU, however, this effort is small compared to the savings that result from the fact that fewer data blocks of the hard disk have to be read or modified. (Of course, compression also saves space on the much faster SSDs, but noticeable speed improvements are not to be expected.) You can certainly use the compressed file system later without the compress option. New or modified files are then no longer compressed, but already existing files remain compressed as long as the files are only read.

The compress option is especially well suited for directories that contain a large number of text files (such as /usr/; the space saved here is almost 50 percent!). On the other hand, the option is not recommended for your user directory if it predominantly contains already compressed files, such as audio, video, PDF, and LibreOffice files. Further compression will then not succeed. The btrfs driver also recognizes this and does not compress the file in question. Nevertheless, this test takes some time.

In reality, for desktop systems it's convenient to use compress for the system partition, but often not for the home partition. Unfortunately, not every distribution allows you to set the mount options during installation.

Needless to say, there are also special cases: for example, if you're running a MySQL database server using the InnoDB table driver that automatically compresses the tables, the MySQL database directory `/var/lib/mysql` should not be in a `btrfs` file system with `compress` option.

18.11.4 Subvolumes

From conventional file systems you know the rule that one partition or one logical volume has one file system. With `btrfs`, it's different. A subvolume is a separate virtual file system. Within the "real" `btrfs` file system, there can be any number of subvolumes. Subvolumes can be treated like directories, but they can also be embedded in the directory tree at any location like file systems.

Subvolumes form a hierarchy according to the directory structure. At the top is the default subvolume, which is automatically created when a `btrfs` file system is set up.

Creating subvolumes

The easiest way to understand this is to take a look at an example. I assume that the `btrfs` file system is located in the `/dev/sdb1` partition and mounted in the `/mnt/btrfs` directory. The `btrfs subvolume create` command now creates two new subvolumes, `sub1` and `data/sub2`:

```
root# btrfs subvolume create /mnt/btrfs/sub1
root# mkdir /mnt/btrfs/data
root# btrfs subvolume create /mnt/btrfs/data/sub2
```

You can use the `btrfs subvolume list` command to get an overview of all subvolumes and snapshots. The three `-a`, `-p`, and `-t` options have the effect that the location of the subvolume within the directory hierarchy is specified exactly, that the parent subvolume is also specified for each subvolume or snapshot, and that the result is formatted as a table:

```
root# btrfs subvolume list /mnt/btrfs/ -a -p -t
ID      gen     parent  top level       path
--      ---     ------  ---------       ----
256     8       5       5               sub1
257     9       5       5               data/sub2
```

When a new `btrfs` file system is created, a default subvolume is automatically set up for the root directory. It always has the ID 5 and is not listed separately. However, in the previous listing, you can see that the new subvolumes were created within this default subvolume (`parent = 5`).

Subvolumes behave almost like ordinary directories. You can create and delete files in them. You can also mount subvolumes like your own file systems at any location in the directory tree. To do this, you can use `mount` with the `subvolid=n` option:

```
root# mkdir /mnt/sub2
root# mount -o subvolid=257 /dev/sdb1 /mnt/sub2
```

The explicit execution of `mount` is required to use subvolumes created in a currently inactive subvolume (i.e., outside the default subvolume with ID=5).

Setting a default subvolume

You can use `btrfs subvolume set-default` to specify the subvolume that will be used by default on the next `mount` command unless a different subvolume is explicitly selected by using the `subvolid` `mount` option. You must pass the volume ID that you previously determined using `btrfs subvolume list` to `set-default`.

Deleting subvolumes

The `btrfs subvolume delete name` command enables you to delete a subvolume, including all files contained in it. The subvolume must of course be detached from the directory tree prior to that:

```
root# umount /mnt/sub2
root# btrfs subvolume delete /mnt/btrfs/data/sub2
```

Note that the memory used by subvolumes is not released immediately with the execution of `btrfs subvolume delete`, but gradually. A kernel process takes care of the necessary cleanup in the background.

Alternatively, you can simply delete subvolumes by deleting the affected directory and its contents (`rm -rf /mnt/btrfs/data/sub2`). However, for subvolumes with many subdirectories and files, this takes much longer.

18.11.5 Snapshots

A *snapshot* is initially a virtual copy of a subvolume. It is *virtual* because no data is copied when a snapshot is created. Not until the original subvolume or the snapshot is modified are two separate branches formed. Snapshots only affect the base subvolume directly, not any existing sub-sub-volumes. Snapshots are often created as a kind of backup of the subvolume in question. Subsequently, the subvolume is then further used and modified, while the snapshot is only used to archive the previous state or to make its contents available to a backup program. But this is only one possible approach. `btrfs` allows changes in the snapshot! The original subvolume and the

snapshot created from it thus become two branches of a start state. (However, btrfs also knows read-only snapshots).

Internally in btrfs, snapshots are treated like subvolumes. Therefore, most subvolume commands of btrfs apply equally to subvolumes and snapshots. The main difference between subvolumes and snapshots is that subvolumes are initially empty, while snapshots contain a virtual copy of the source directory.

btrfs versus LVM

The term *snapshot* is used completely differently in btrfs and LVM (see Section 18.16). In LVM, a snapshot is an immutable image of a logical volume. When you create the snapshot, you must specify the maximum amount of disk space that the snapshot can consume. This memory space is used to archive data blocks of the original LV, if necessary, before they are changed. When the snapshot space is used up, the snapshot becomes invalid and can no longer be used.

With btrfs, on the other hand, the contents of the snapshot are mutable. A snapshot is therefore a new branch of a directory tree that is identical to the snapshot at the time it is created. From this point on, both branches can develop independently of each other. The two branches take up more memory the more files are changed. btrfs simply uses the btrfs memory pool to store the changes. This works until the capacity of the entire file system is exhausted. btrfs snapshots thus provide a lot more features and flexibility than LVM snapshots!

Example

The starting point for the following example is a btrfs file system in the /dev/sdb1 partition. The file system is integrated into the directory tree at the /mnt/btrfs location. btrfs subvolume snap then creates a snapshot of the entire file system. At the same time, the /mnt/btrfs/snap1 directory is created. The snapshot can be used via this directory or integrated like a separate file system in the directory tree via mount. Note that you must use the mount option option of subvol=name that you already know from the previous section:

```
root# btrfs subvolume snapshot /mnt/btrfs/ /mnt/btrfs/snap1
root# mkdir /mnt/snap1
root# mount -o subvol=snap1 /dev/sdb1 /mnt/snap1/
```

You can now independently create, modify, and delete files in both the original file system and the snapshot (i.e., without affecting each other).

Typically, btrfs snapshots are mutable. If you want a read-only snapshot for backups, you can run btrfs subvolume snapshot with the -r option.

18.11.6 Distributing btrfs File Systems across Multiple Devices (RAID)

The `btrfs` driver can distribute file systems across multiple hard disks or devices and supports RAID levels 0, 1, 5, 6, and 10 without resorting to the usual Linux RAID driver, `mdadm`.

Adding a device

The simplest case of multidevice file systems occurs mostly when a `btrfs` file system becomes too small. You can then easily add another device (i.e., an empty disk partition or an unused logical volume). This increases the size of the file system accordingly. You don't need to reformat a partition or explicitly resize the file system; `btrfs` does all these tasks on its own:

```
root# btrfs device add /dev/sdb2 /mnt/btrfs
```

Initially, all data is then located on the first device, while the second device is only used gradually. If the devices are located on different physical hard disks (and only then!), you will achieve a speed gain by distributing the existing files across all devices using `btrfs filesystem balance`:

```
root# btrfs filesystem balance /mnt/btrfs
```

Note, however, that running `btrfs filesystem balance` takes a long time and is rarely worth the effort. New or changed files are automatically distributed across both devices anyway.

It's also possible to remove a device again. The data stored on the device is then transferred to the other devices first, which is why the execution of the following command can take a very long time:

```
root# btrfs device delete /dev/sdb1 /mnt/btrfs/
```

mkfs.btrfs with multiple devices

You can also set up a `btrfs` file system with multiple devices from the start by passing multiple devices to `mkfs.btrfs`:

```
root# mkfs.btrfs /dev/sdb1 /dev/sdc1
```

By default, the file system metadata is then duplicated (equivalent to RAID-1), but the actual data is distributed across all devices. The metadata contains the management information of the file system, such as inode lists as well as trees for searching files. Unfortunately, the resulting file system is then neither optimal in terms of speed (it's slower than a RAID-0 system because of the duplication of metadata) nor able to keep up with RAID-1 in terms of security because the actual data is not stored redundantly.

When you run `mount`, you specify any device of the file system. After a computer reboot, the `btrfs device scan` command must be executed so that

btrfs knows which devices with btrfs file systems exist and how they are associated with each other:

```
root# mount -t /dev/sdb1 /mnt/btrfs
```

You can find out which RAID variant is used for which type of data by using btrfs filesystem df:

```
root# mkfs.btrfs /dev/sdb1 /dev/sdc1
root# mount /dev/sdb1 /mnt/btrfs
root# cp -a /etc /mnt/btrfs
root# btrfs filesystem df /mnt/btrfs
Data, single:          total=1.01GiB,   used=14.28MiB
System, DUP:           total=8.00MiB,   used=16.00KiB
Metadata, DUP:         total=511.94MiB, used=1.86MiB
GlobalReserve, single: total=16.00MiB,  used=0.00B
```

This (somewhat abbreviated) output means that the data is distributed across both devices (RAID-0), but internal file system data (system and metadata) is mirrored. If a hardware failure occurs on either medium, duplicates of at least all metadata are available.

RAID-0

If you want to create a "real" RAID system where data and metadata are handled consistently, you need to pass the desired RAID level to mkfs.btrfs with -d (for the data) and -m (for the metadata). In addition, you need to pass the desired number of devices to mkfs.btrfs. If mkfs.btrfs detects that a partition already contains a file system, you must also specify the -f option (*force*) to overwrite that file system. The following command creates a RAID-0 system (striping):

```
root# mkfs.btrfs -f -d raid0 -m raid0 /dev/sdb1 /dev/sdc1
```

RAID-1

A RAID-1 file system is set up in a similar way by using the following command:

```
root# mkfs.btrfs -d raid1 -m raid1 /dev/sdb1 /dev/sdc1
```

It gets interesting when a device fails. To test this case, I removed the /dev/sdc disk. To be able to use the file system, the additional mount option degraded must now be used:

```
root# mount -o degraded /dev/sdb1 /mnt/btrfs
```

To restore the RAID array, you must add a new device of the same size as the file system. In the following example, this is again /dev/sdc1, but this partition is now from a new hard disk. To redistribute the file system across both

devices and thus restore RAID-1 redundancy, you must also run `btrfs filesystem balance`. For large file systems, the execution of this command naturally takes a long time. The file system can be used during this process, albeit at greatly reduced speed:

```
root# btrfs device add /dev/sdc1 /mnt/btrfs
root# btrfs filesystem balance /mnt/btrfs
```

Only now can the defective device be removed from the file system. You can use the `missing` keyword to specify the device:

```
root# btrfs device delete missing /mnt/btrfs
```

Finally, you can now omit the `degraded` option from the next `mount` command.

18.11.7 Determining the Use of a btrfs File System (df)

For other file systems, you can easily use `df -h` to determine how much space is available, how much of it is used, and how much is still free. However, for `btrfs` file systems, `df` sometimes returns misleading results, especially in the context of RAID. The reasons for this are the separation between system data, metadata, and the actual files; the use of different RAID levels for data and metadata; possibly active compression functions; and the use of pools that `btrfs` reserves in advance for different purposes.

In addition to `df`, you can also run the `btrfs` commands `filesystem show` and `filesystem usage`. The following example should help you to at least correctly interpret the data that `btrfs` provides. The starting point is a small `btrfs` RAID-1 system consisting of two 10 GiB partitions. The entire `/usr` directory of the test system was copied to this file system (space requirement according to `du` is approximately 4.6 GiB):

```
root# mkfs.btrfs -d raid1 -m raid1 /dev/sdb1 /dev/sdc1
root# mount /dev/sdb1 /mnt/btrfs
root# cp -a /usr /mnt/btrfs
```

df The result of `df` is almost perfect here despite the use of RAID-1. This is a big improvement over older `btrfs` versions where `df` was way off in the same scenario:

```
root# df -h /mnt/btrfs/
Filesystem   Size  Used  Avail.  Used%  Mounted on
/dev/sdb1     10G  4.5G   4.8G   49%    /mnt/btrfs
```

`btrfs filesystem show` tells you that the file system is composed of two 10 GiB devices and that both devices are filled to the same extent: **filesystem show**

```
root# btrfs filesystem show  /dev/sdb1
Label: none  uuid: e4f2c9d3-5b88-4019-af19-682a3417d8e6
        Total devices 2 FS bytes used 4.43GiB
        devid    1 size 10.00GiB used 6.01GiB path /dev/sdb1
        devid    2 size 10.00GiB used 6.01GiB path /dev/sdc1
```

`btrfs fi usage` provides detailed information about how the reserved data is used: **fi usage**

```
root# btrfs fi usage /mnt/btrfs/
Overall:
    Device size:                  20.00GiB
    Device allocated:             12.02GiB
    Device unallocated:            7.98GiB
    Device missing:                  0.00B
    Used:                          8.87GiB
    Free (estimated):              4.72GiB      (min: 4.72GiB)
    Data ratio:                       2.00
    Metadata ratio:                   2.00
    Global reserve:               16.00MiB      (used: 0.00B)
Data,RAID1: Size:5.00GiB, Used:4.27GiB
   /dev/sdb1       5.00GiB
   /dev/sdc1       5.00GiB
Metadata,RAID1: Size:1.00GiB, Used:169.08MiB
   /dev/sdb1       1.00GiB
   /dev/sdc1       1.00GiB
System,RAID1: Size:8.00MiB, Used:16.00KiB
   /dev/sdb1       8.00MiB
   /dev/sdc1       8.00MiB
Unallocated:
   /dev/sdb1       3.99GiB
   /dev/sdc1       3.99GiB
```

In summary, this means that `btrfs` has reserved 5 GiB each for storing files on both partitions, 0.73 GiB of which is still free. Another 1 GiB has been reserved for metadata, but so far only 169 MiB of this is in use. Finally, there is just under 4 GiB of free space on each of the two partitions.

Disk-Full Problems

You should size `btrfs` systems as generously as possible! `ext4` file systems can be written to almost to the last byte (even though this is not recommended).

btrfs, however, causes a much larger overhead for subvolumes, snapshots, and the like compared to ext4. For this reason, you should allow for more space than with other file system types to ensure smooth operation.

18.11.8 btrfs Configuration in openSUSE

No other Linux distribution uses btrfs as extensively by default as SUSE. Current versions of openSUSE or the SUSE Enterprise distributions use btrfs for the system partition and xfs for the home partition. Within the system partition, YaST creates various subvolumes during the installation for directories such as /var/xxx, /opt, and /srv. When you install openSUSE on a relatively small volume, /home is also integrated into the btrfs file system.

In addition, YaST creates two snapshots for each administrative task: a pre snapshot (before) and a corresponding pre-post snapshot (after). The benefit of this configuration is that you can restore your system to an older state in the YaST Filesystem Snapshot module (formerly called *Snapper*; see Figure 18.3) without also making a leap into the past with your personal data (/home directory). Alternatively, you can reset only individual files. If there are boot problems, you can even select an older snapshot in GRUB and boot the system in this state.

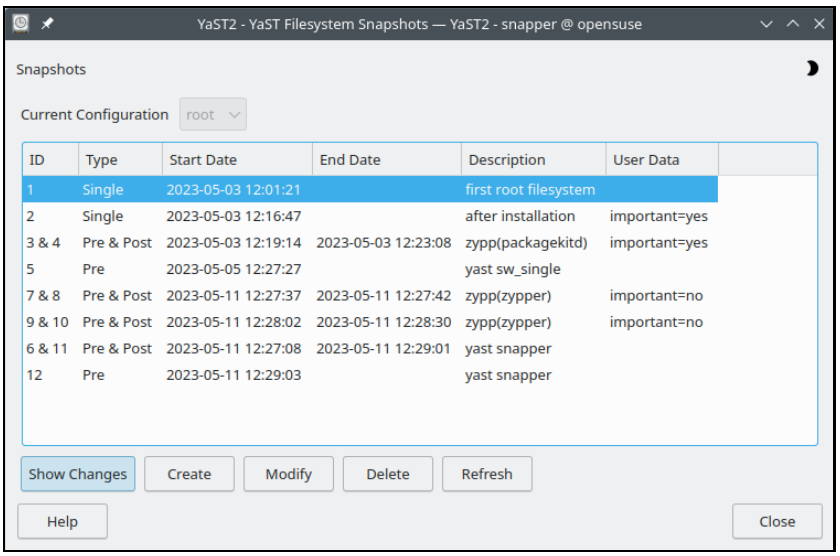


Figure 18.3 Managing Snapshots in YaST

The price for this security against admin errors is a btrfs system in which you can get dizzy from all the subvolumes and snapshots:

```
root# btrfs subvol list / -p -a  -t
ID      gen     parent  top level   path
--      ---     ------  ---------   ----
256     32      5       5           <FS_TREE>/@
258     7705    256     256         <FS_TREE>/@/var
259     7158    256     256         <FS_TREE>/@/usr/local
260     7704    256     256         <FS_TREE>/@/tmp
261     46      256     256         <FS_TREE>/@/srv
262     7703    256     256         <FS_TREE>/@/root
263     35      256     256         <FS_TREE>/@/opt
264     7704    256     256         <FS_TREE>/@/home
265     26      256     256         <FS_TREE>/@/boot/grub2/x86_64-efi
266     4376    256     256         <FS_TREE>/@/boot/grub2/i386-pc
267     7169    256     256         <FS_TREE>/@/.snapshots
268     7705    267     267         <FS_TREE>/@/.snapshots/1/snapshot
...
```

Configuration and administration

The configuration files for the Filesystem Snapshots system are located in `/etc/snapper`. From `/etc/snapper/configs/root`, it appears that snapshots for the last ten YaST actions are backed up by default. Older snapshots are automatically deleted. Optionally, you can also enable hourly snapshots in this file, which also includes the automatic deletion of older snapshots.

Instead of YaST, you can also use the command of the same name for Snapper configuration:

- `snapper list` lists all available snapshots.
- `snapper create` creates a new snapshot.
- `snapper delete n` deletes the specified snapshot.
- `snapper delete n1-n2` deletes the snapshots with numbers between `n1` and `n2`.
- `snapper cleanup type` deletes old snapshots, with several cleanup algorithms to choose from: `number`, `timeline`, or `empty-pre-post`.
- `snapper undochange n1..n2 [files]` undoes the changes made between snapshot `n1` and snapshot `n2` either for all affected files or only for the files specified as parameters.
- `snapper rollback [n]` makes the specified or the most recent available snapshot the new default subvolume.

Determining space requirements

We've already dealt with the question of how much storage space is still free in a `btrfs` file system. But maybe you want to know how much disk space a single subvolume or snapshot takes up. To answer this question, you must first enable the quota features of `btrfs`:

```
root# btrfs quota enable /
```

After a while (usually after a few minutes), `btrfs qgroup show` then shows which subvolume or snapshot is taking up how much space:

```
root# btrfs qgroup show -e --human-readable --sort=qgroupid  /
qgroupid         rfer         excl     max_excl
--------         ----         ----     --------
0/5          16.00KiB     16.00KiB         none
0/256        16.00KiB     16.00KiB         none
0/258       260.24MiB    260.24MiB         none
0/259        16.00KiB     16.00KiB         none
...
0/356         5.46GiB     48.00KiB         none
0/357         5.46GiB     48.00KiB         none
0/358         5.48GiB      2.57MiB         none
1/0           6.83GiB      1.48GiB         none
```

Unfortunately, the result is difficult to read in this format. First, the subvolumes or snapshots are not designated by their names, but only by their ID numbers. Second, I still owe you the explanation of what the difference is between the `rfer` column and the `excl` column: `rfer` specifies the total memory requirement of a subvolume/snapshot, while `excl` specifies the extent to which this memory is exclusively used by the subvolume/snapshot in question. So this storage is not shared with other subvolumes/snapshots.

Disabling Snapper

Three actions are required to disable automatic snapshots:

- **YaST snapshots during admin tasks**
  YaST is responsible for creating these snapshots. Accordingly, there is an option for this in the YaST configuration file, which can be set to `yes` or `no`:

  ```
  # file /etc/sysconfig/yast2
  USE_SNAPPER="no"
  ```

- **Snapshots during package installations (Zypper)**
  A Zypper plugin is responsible for creating snapshots during the installations of software or updates. You will need to uninstall that plugin:

  ```
  root# zypper remove snapper-zypp-plugin
  ```

- **Hourly snapshots**
  Hourly snapshots are turned off by default on openSUSE. To enable or disable, you simply need to set the `TIMELINE_CREATE` variable to `yes` or `no`:

  ```
  # file /etc/snapper/configs/root
  TIMELINE_CREATE="no"
  ```

Making room

If the root file system is already bursting at the seams, you can use `snapper delete` or YaST to delete snapshots that are no longer required and `zypper clean` to clear the cache of the package management system.

Recommendations for SUSE Installations

For private users of openSUSE, I recommend using the `ext4` file system for the system partition instead of `btrfs` during installation.

The benefits associated with `btrfs` come with such complexity and hard-to-predict side effects that only `btrfs` experts can help with problems.

Personally, I also avoid `btrfs` in professional use—but admittedly, you may have a different opinion here. So if you decide to use `btrfs`, be sure to size the system partition twice as large as in conventional installations!

18.11.9 btrfs Configuration in Fedora

Since version 33, Fedora uses `btrfs` by default for desktop installations. However, the configuration (last tested on Fedora 38) is much simpler than on SUSE: by default, only three subvolumes are created for `/root`, `/home`, and `/var/lib/machines`. The only real `btrfs` feature activated is automatic compression.

The Fedora setup is very clear, but it has the disadvantage that handling images and large database files is not as efficient as on SUSE because of the always active COW. If necessary, you must set up subvolumes without COW for the data directories themselves *before* installing virtualization or database systems.

compsize

In case you are wondering how much you will benefit from automatic compression, the `compsize` command provides answers (although the following listing is slightly abbreviated):

```
root# compsize -x /
Type       Perc     Disk Usage   Uncompressed Referenced
TOTAL       60%       5.4G         9.0G          10G
none       100%       3.5G         3.5G         3.9G
zstd        34%       1.8G         5.4G         6.7G
```

According to this, `btrfs` has dispensed with compression for files with a volume of 3.5 GiB (`none` line). Other files of 5.4 GiB were compressed and therefore consume only 1.8 GiB on the disk (`zstd` line). In total, the entire file system takes up only about 60% of the space it would require without compression (`TOTAL` line).

The difference between the Uncompressed and Referenced columns concerns the handling of files that are empty in wide areas and contain only 0 bytes. Not only btrfs but also most other Linux file systems recognize such *sparse files* and deal with them efficiently. The Referenced column indicates how much space the files would take up on a file system that is not optimized in this respect.

Snapshots

Fedora does not use snapshots by default. However, the snapper tool familiar from SUSE can also be installed on Fedora—for example, to create a snapshot with every dnf action. The installation instructions can be found at the following address: *https://sysguides.com/install-fedora-38-with-snapshot-and-rollback-support*.

18.12 The xfs File System

The xfs file system was developed by SGI for their workstations with the Unix-like operating system IRIX. Later, the file system was ported for Linux and gradually improved again and again. The file system is considered to be mature, stable, and especially efficient in handling very large files. This also applies to file systems that are larger than 16 TiB.

Since 2014, xfs has been used as the default file system by both RHEL and current SUSE distributions. (SUSE differentiates between the system partition and the data partitions; for the system partition, btrfs is provided, for the data partitions xfs). You can find more information about the xfs file system at *https://xfs.wiki.kernel.org*.

/etc/fstab

Entries for an xfs file system in /etc/fstab usually look like the following example. Additional mount options are needed only very rarely. They are listed in man mount:

```
# /etc/fstab
/dev/sdb13      /data          xfs  defaults 0 0
```

Setting up an xfs file system

To set up an xfs file system in a partition, you simply need to run mkfs.xfs. If the command is not available, you must first install the xfsprogs package. The parameters output by mkfs.xfs can be read later via xfs_info if required:

```
root# mkfs.xfs /dev/sdb1
meta-data=/dev/sdb1    isize=256    agcount=4, agsize=1048512 blks
         =             sectsz=512   attr=2, projid32bit=1
         =             crc=0        finobt=0
data     =             bsize=4096   blocks=4194048, imaxpct=25
```

```
naming   =version 2    bsize=4096   ascii-ci=0 ftype=0

log      =internal log bsize=4096   blocks=2560, version=2
         =             sectsz=512   sunit=0 blks, lazy-count=1
realtime =no           extsz=4096   blocks=0, rtextents=0
```

Now all that's missing is a `mount` command, and you're ready to use the file system:

```
root# mount -t xfs /dev/sdb1 /test
```

Checking the file system

The integrity of `xfs` file systems is automatically checked during each `mount` operation. However, only the journaling log is analyzed. For a manual check, you must run `xfs_check`. This is only possible if the file system is not mounted. If the command detects errors, you can try to fix them using `xfs_repair`.

To ensure compatibility with the other file systems, the `fsck.xfs` command can be used. However, this command does not perform any task and always returns OK as a result.

Changing parameters of the file system

`xfs_growfs` enlarges an `xfs` file system during operation. For this to work, the file system must be mounted! The command assumes that the underlying data partition was previously enlarged. The name of the command already suggests it: an `xfs` file system can only be enlarged using `xfs_growfs`, not reduced. (Functions to shrink XFS file systems are being worked on.)

`xfs_admin` changes various parameters of the file system, such as the name (label) and the UUID number. The file system must be removed from the directory tree upfront (with `umount`).

18.13 Windows File Systems (vfat and ntfs)

Some Linux users have a Windows version installed on their computer in parallel. But external data media also often use Windows file systems (USB flash drives, memory cards of digital cameras, etc.). In the following sections, you'll learn how you can access Windows file systems on Linux—no matter whether they are located in a partition of the internal hard disk or on an external data medium.

Variants

There are three fundamentally different Windows file systems:

- **FAT, VFAT, and exFAT**
  There are countless variants of the FAT file system. The oldest versions historically are FAT12 for floppy disks, FAT16 for file systems up to 2 GiB,

and FAT32 for file systems up to 8 TiB and files up to 4 GiB. Windows 95 also introduced VFAT, which finally allowed file names of more than 8+3 characters. The combination of FAT32 and VFAT is most commonly used today, such as on many SD cards up to 32 GiB.

The FAT variant exFAT was developed especially for large flash cards (e.g., for digital cameras). exFAT allows files up to 16,777,216 TiB in size and supports ACLs and transactions.

- **New Technology File System (NTFS)**
  NTFS was introduced with Windows NT and is used by all current Windows versions. Compared to FAT, NTFS offers a higher degree of security (access rights, journaling, etc.) as well as various additional functions. The file system size is almost unlimited at 16,777,216 TiB.
- **ReFS**
  Since Windows 8 or Windows Server 2012, ReFS can also be used as an alternative to NTFS. This new file system is more modern than NTFS, but it also has some limitations. It hasn't yet achieved widespread use and is not used in standard desktop installations. This might change with the upcoming Windows 12 OS.

Linux support

Almost all Linux distributions can read and write (V)FAT and NTFS file systems right away. In the past, exFAT support was somewhat more problematic. But because Microsoft gave its consent in 2019 to allow the driver to be integrated into the kernel, it is now available by default in many modern distributions. There is no open-source driver for ReFS yet, but there is a proprietary driver from the Paragon company (*https://www.paragon-software.com/business/refs-linux*).

Setting up a Windows file system

It's rarely necessary to set up Windows file systems on Linux, but it is of course possible. Before you can run `mkfs.exfat` or `mkfs.ntfs`, you may need to install the `exfat-utils` or `ntfsprogs` package:

```
root# mkfs.vfat -F 32  /dev/sdb1  (VFAT)
root# mkfs.exfat       /dev/sdb1  (exFAT)
root# mkfs.ntfs --fast /dev/sdb1  (NTFS)
```

Be Sure to Avoid Data Loss when Accessing Windows File Systems!

On a dual-boot machine running either Linux or Windows, writes performed on Linux can result in an inconsistent Windows file system and data loss. Windows is not shut down completely by default for speed reasons. Instead, a quick start function is active, which is based on Windows being put into a special hibernation state.

Recent changes to the file system are saved in a special file that Linux does not see. You should always deactivate the quick start function if you are running a mixed Windows/Linux setup. You can find relevant instructions on this at *https://pureinfotech.com/disable-fast-startup-windows-11-10*.

To shut down Windows completely only once, you must first close all programs and then execute the `shutdown /p` command in `cmd.exe`.

The easiest way to avoid these problems is to synchronize shared Windows/Linux data externally, such as in a cloud directory.

Access rights

VFAT does not know the concept of access rights at all. NTFS does support access rights, but not in the typical Unix/Linux way. This results in a problem: Which Linux user has what access rights for Windows files? The answer is given by the `mount` options `uid`, `gid`, and `umask/fmask/dmask`. These options set the owner, group membership, and access bits for the Windows file system uniformly for all files of this file system and independent of any NTFS access rights.

18.13.1 The VFAT File System

Default settings

The `vfat` driver detects the FAT type (FAT12/-16/-32) independently. On Linux, Windows file names are represented in the Latin-1 (ISO 8859-1) character set. By default, the user running `mount` is allowed to read and write all files and directories; all other users are allowed to read everything, but not to modify anything. Of course, you can change these settings by way of options.

VFAT in /etc/fstab

A typical entry in `/etc/fstab` for a local VFAT partition on the hard disk looks as follows:

```
# /etc/fstab
/dev/sda1      /media/win1  vfat    utf8,uid=1000,noexec  0 0
```

This ensures that the user with the user number 1000 can change all files and that special characters in Windows file names are displayed as UTF8 characters on Linux. For security reasons, you must not run programs stored in the Windows file system (`noexec` option).

The following `fstab` line does not automatically mount the Windows partition in the directory tree (`noauto`). Thanks to `users`, however, any user may run `mount`. `gid=users` causes the group membership of the Windows files to be determined by the user's default group (and not the currently active group):

```
/dev/sda1      /media/win1  vfat    noauto,users,gid=users,utf8  0 0
```

18.13.2 The NTFS File System

The ntfs driver of Linux supports read and write access and can handle streams. However, the driver cannot read or write encrypted files and cannot create compressed files (although it can read them).

Unlike most other file system drivers, ntfs is not implemented as a kernel module, but as a FUSE driver. This is a kernel module that communicates with external programs. FUSE thus allows the actual file system driver to be implemented outside the kernel.

NTFS in /etc/fstab

To grant the user with the UID 1000 write access to an NTFS disk, you need an fstab line, as shown in the following pattern:

```
# /etc/fstab
/dev/sda1    /media/win   ntfs   uid=1000,gid=1000  0 0
```

Streams

Streams are a special feature of the NTFS file system, and an NTFS file can consist of multiple streams. Each stream has the same function as a conventional file. The default stream is automatically read or changed during a normal file access. For the ntfs driver, the streams_interface option controls access to streams. In the default xattr setting, streams are considered like file attributes. Streams are accessed using the get- or setfattr commands from the attr package (see Chapter 9, Section 9.7). getfattr -d -e text returns a list of all attributes, displaying their contents as text:

```
root# mount /dev/sda1 /media/win
root# cd /media/win
root# cat > streamtest
abc <Ctrl>+<D>

root# setfattr -n user.stream1 -v "efg" streamtest
root# setfattr -n user.stream2 -v "xyz" streamtest
root# cat streamtest
abc
root# getfattr -d -e text streamtest
# file: streamtest
user.stream1="efg"
user.stream2="xyz"

root# cd
root# umount /media/win
```

Alternatively, you can also use streams_interface=windows. This setting activates the typical Windows notation in the form filename:streamname:

```
root# mount -o streams_interface=windows /dev/sda1 /media/win
root# cd /media/win
root# cat streamtest
abc
root# cat streamtest:stream1
efg
```

The ntfsprogs package contains various commands that support the administration of NTFS file systems. Table 18.7 gives an overview. **Administration**

Command	Meaning
mkfs.ntfs	Sets up an NTFS file system
ntfsclone	Copies an NTFS file system
ntfsinfo	Provides information about an NTFS file system
ntfslabel	Assigns a name to an NTFS partition
ntfsresize	Changes the size of the NTFS file system
ntfsundelete	Tries to restore deleted files

Table 18.7 Commands of ntfsprogs Package

18.14 Swap Partitions and Files

If there isn't enough memory to run all programs, and swap partitions or files are available, then Linux swaps parts of the memory there (*paging*). This way, Linux can use more memory than is available in RAM.

Setting up a swap partition is usually done as part of the installation. You can check whether and how much swap space is available or actually used by using the free command. In the example ahead, 1,519 MiB of RAM and 2,000 MiB of swap memory are available. Of the RAM, 401 MiB is currently used for programs and data, and the rest is used as a buffer or cache for files. The swap memory is currently unused:

```
root# free -m
             total       used       free     shared    buffers     cached
Mem:          1519       1479         39          0         67       1010
-/+ buffers/cache:        401       1117
Swap:         2000          0       2000
```

If a computer is running for a longer time, it will use the swap memory even if a lot of RAM is available. Here's why: The kernel manages a cache for read

accesses to files. If a file is needed again later, it can be read from the cache. As soon as the cache is larger than the otherwise free RAM, Linux swaps out memory blocks that have not been used for a long time to the swap partition. This is by no means a sign that too little RAM is available. Linux simply tries to use the available memory as efficiently as possible.

/etc/fstab

The following lines show two entries for swap partitions in `/etc/fstab`. The `pri` option causes the two partitions to be treated equally by Linux. This provides a speed increase as with striping or RAID-0 (see Section 18.15), provided that the partitions are located on two separate hard disks. If there is only one swap partition, you should just specify `defaults` instead of `pri=0`:

```
# /etc/fstab: Swap partitions
/dev/sda9   swap   swap    pri=1   0 0
/dev/sdb7   swap   swap    pri=1   0 0
```

Controlling the swap behavior

When RAM runs out of memory, the Linux kernel uses a relatively complex algorithm to decide whether to free cache memory in favor of other memory requests or to swap out recently unused memory areas to the swap partition. Using the `/proc/sys/vm/swappiness` kernel parameter, you can define yourself whether the kernel should shrink the cache or swap data if possible. To learn what kernel parameters are and how to set them, see Chapter 21, Section 21.7.

The default setting for `swappiness` is 60; the possible value range is from 0 to 100. The value 0 means that the kernel avoids paging if possible, while 100 means that memory that has been unused for a long time ends up in a swap partition as soon as possible. More details about the `swappiness` parameter can be found at *https://lwn.net/Articles/83588*.

In practice, you're most likely to notice swap behavior if you leave your computer running overnight and a program accesses many files on the hard disk in your absence, such as a backup script or a program to create a search index. Because of the large number of file accesses, the cache grows a lot. Using `swappiness=60` or an even higher value, the kernel will then swap out memory that has not been used for hours. For example, this could affect the memory pages of Firefox or GIMP. If you want to continue using GIMP the next day, it will take a few seconds to transfer those pages from the swap partition back into the memory. By using `swappiness=0`, you avoid this wait time.

How much swap space?

In the past, the recommendation was to provide memory equal to approximately twice the amount of RAM as swap memory. With increasing RAM size, however, this rule of thumb is no longer useful. If you use Linux

primarily as a desktop system, a much smaller swap partition is quite sufficient (such as 2 GiB of swap space with 16 GiB of RAM). The swap memory only needs to be larger than the RAM if you want to use hibernation mode on your notebook computer. I wouldn't take that into account, though; I've had mostly bad experiences with hibernation and only use standby mode now.

Linux without a swap partition

Again and again, the question comes up as to whether you can or should completely do without a swap partition if you have a lot of RAM. Although opinions differ, a sensibly sized swap space will generally increase the efficiency of Linux. If you use Linux as a desktop system on a modern, well-equipped notebook computer, the efficiency improvement will be so small that you can hardly measure it. On a server or on a computer that is used intensively for software development or other memory-intensive applications, you are simply giving away computing power without swap space.

If you want to dig deeper into the pros and cons of a swap partition, I recommend a great blog article available at *https://chrisdown.name/2018/01/02/in-defence-of-swap.html*.

Swap files

Swap files are not quite as efficient as swap partitions but are easier to resize. (Unfortunately, automatic resizing like you'd find on Windows or macOS is not possible.) Currently, Ubuntu is the only major distribution that sets up a swap file by default for desktop installations.

To create a new swap file yourself, you can use the `dd` command to create an empty file with a specified size. Here, `/dev/zero` is used as the data source. The size specification results from the product of the `bs` and `count` parameters, here 1 GiB. Then the swap file can be formatted like a swap partition using `mkswap` and activated via `swapon`:

```
root# dd bs=1M if=/dev/zero of=/swapfile count=1024
root# mkswap /swapfile 1000
root# sync
root# swapon -v /swapfile
swapon on device /swapfile
```

Note that the `btrfs` file system does not support swap files!

To resize a swap file, you must first disable it using `swapoff` and then set up a new file of the size you want.

Swap files are included in `fstab` like swap partitions:

```
# extension to /etc/fstab
/swapfile  none      swap          sw       0  0
```

18.15 RAID

The basic idea behind *Redundant Array of Independent Disks* (RAID) is to logically link partitions of multiple hard disks or SSDs. The goal here is to create a more reliable overall system through redundancy.

Hardware versus software RAID

There are two basic ways to implement RAID: through hardware (i.e., a controller that executes the RAID logic itself) or through software executed by the computer's CPU. Hardware RAID is mainly used in expensive server systems. Its greatest advantages are that the RAID controller acts independently of the operating system.

With regard to software RAID, a distinction is made among different types, depending on where the software comes from:

- **Linux software RAID**
  Linux can combine multiple disks into a RAID. From a Linux perspective, this RAID variant is preferable. If this book talks about RAID without further explanation, it always means Linux software RAID!
- **Windows software RAID**
  Windows also supports different RAID variants in the form of software RAID. Windows file systems set up in this way are not readable by Linux.
- **Host RAID (fake RAID)**
  In this variant, the BIOS or EFI implements different RAID levels in combination with an operating system driver. The derogatory term *fake RAID* is explained by the fact that in the past many RAID controllers were advertised as if they were real hardware RAID controllers. Today, fake RAID is hardly ever used.

RAID level

There are several methods to join partitions. These variants are referred to as *RAID levels*:

- **RAID-0 (striping)**
  In RAID-0, multiple physical partitions are combined into one larger partition. Here, the data is distributed in parallel in small blocks to the individual partitions so that the data is read alternately from all disks when accessed. Ideally, this results in a multiplication of the data rate—a tripling, for example, in the case of three hard disks. Such speed increases are more difficult to achieve with SSDs because the maximum speed is often limited by the bus system. Apart from that, the speed gain is often much smaller than hoped for and only comes into play with large files.

  RAID-0 has a serious disadvantage: the risk of failure increases because *one* defective hard disk or SSD leads to the loss of *all* data.

- **RAID-1 (mirroring)**
  With RAID-1, the same data is stored on two hard disks or SSDs. If one disk fails, all data is still available on the other device. The advantage is higher safety, while the disadvantage is halved capacity. RAID-1 does not provide any speed advantages.
- **RAID-10**
  RAID-10 combines RAID-1 and RAID-0 and requires at least four disks: disks 1 and 2 form a RAID-1 array while disks 3 and 4 form another RAID-1 array. At the next level, the two RAID-1 arrays are combined into a RAID-0 array. RAID-10 thus combines the advantages of RAID-0 (speed) and RAID-1 (security).
- **RAID-5 (parity striping)**
  Basically, RAID-5 works like RAID-0, but additional parity information is stored in a partition, which changes for each data block. If a disk fails, the entire data can be reconstructed. However, the failure of two or more disks will still result in a complete loss of data. RAID-5 requires at least three disks.

  RAID-5 is as secure as RAID-1 and about as fast as RAID-0 for read accesses. In addition, RAID-5 has the advantage that the data portion required for redundancy becomes smaller with the number of hard disks: with RAID-1, the capacity loss is always 50%, but with RAID-5, it is only 33% with three hard disks, 25% with four, 20% with five, and so on.

  However, RAID-5 also has disadvantages compared to RAID-1. First, write operations are slower than in RAID-1, especially when small amounts of data change frequently. The reason is that even for small changes, the parity information for an entire block of data has to be recalculated and stored. Second, after replacing a defective disc, the reconstruction of the RAID-5 array takes a very long time, much longer than with RAID-1.
- **RAID-6**
  RAID-6 works like RAID-5, but it's double redundant and requires at least four hard disks. Even if two hard disks fail, there is no data loss.

Information about more RAID levels as well as many interesting details and basics of RAID can be found at *https://en.wikipedia.org/wiki/RAID*.

This section deals exclusively with the administration of Linux software RAID based on `mdadm`. For space reasons, I will only describe RAID levels 0 and 1.

If you use the `btrfs` file system and want to use RAID levels 0, 1, 5, 6, or 10, you can do without the RAID functions of the multidevice driver described here. `btrfs` contains RAID functions itself.

Server deployment In real life, RAID is used almost exclusively on servers. The primary motivation is to protect against data loss in the event of defective hardware. RAID-1 is typically used for "small" servers. Most data centers where you can rent Linux servers help with configuration tools or scripts for setting up RAID-1. A RAID configuration on a notebook or desktop computer is theoretically possible, but nowadays completely uncommon.

RAID and LVM LVM can be combined with RAID by using a RAID array as the basis for LVM. In this case, RAID must be configured first, followed by LVM.

18.15.1 Manual Configuration Using mdadm

As mentioned earlier, RAID setup usually takes place during the Linux installation. Nevertheless, in this section I'll show you the steps necessary to set up software RAID manually. If you plan to use software RAID on a "real" server, you should definitely familiarize yourself with the basic principles and with the `mdadm` command before that. The easiest way to do this is in a virtual machine. Once you have set up a Linux distribution of your choice, you can add two or more virtual disks. These will then be your playground for learning RAID.

mdadm Unless you have already set up a RAID array during the installation, you must install the `mdadm` package, which contains the RAID administration command of the same name. In some distributions, the package management command suggests that you also install a mail server. When the RAID system detects a hardware defect, it can then automatically send an email to the administrator.

If you have not yet dealt with the subject of the mail server or if you only want to try out RAID in a virtual machine, you can do without the mail server. For this reason, on Debian and Ubuntu, you should use the `--no-install-recommends` option when using `apt` for the installation.

md_mod Within Linux, the so-called multidevice driver is responsible for software RAID. In some distributions, this driver is integrated directly into the kernel; otherwise the `md_mod` kernel module (formerly simply `md`) will be loaded automatically during system startup. `dmesg` should contain appropriate messages in any case. You should also make sure that the `/proc/mdstat` pseudofile exists. It provides information about the current state of the RAID system.

`md_mod` places a logical layer between the hard disk or SSD access driver and the file system driver (e.g., `ext4`). In addition, `md_mod` forms a new logical device from multiple hard disk partitions that can be accessed by the file system driver (`/dev/mdN`). After the RAID configuration, you no longer use a

hard disk partition directly, but a RAID partition—/dev/mdN—to set up your file system on.

mdadm.conf

The central configuration file is /etc/mdadm.conf or /etc/mdadm/mdadm.conf. This file should contain data about all active RAID arrays in addition to some global RAID settings. You can create a completely new configuration file using /usr/share/mdadm/mkconf. This is useful if the configuration file has been lost or if you are working on a live or rescue system.

The usual configuration procedure is unusual. First, you set up or modify the required RAID arrays by executing mdadm commands. Then you extend the predominantly existing mdadm.conf file based on the configuration now present. You can determine the key data of the active RAID arrays using mdadm --examine --scan, and you add them to the existing configuration file by using >>:

```
root# mdadm --examine --scan >> /etc/mdadm/mdadm.conf
```

If mdadm.conf contained RAID definitions before, you must remove them using an editor so that no array is defined twice. The following lines show an example of the structure of mdadm.conf:

```
# file /etc/mdadm/mdadm.conf oder /etc/mdadm.conf
DEVICE partitions
CREATE owner=root group=disk mode=0660 auto=yes
HOMEHOST <system>
MAILADDR root
ARRAY /dev/md0 level=raid1 num-devices=2 UUID=36c426b0:...
ARRAY /dev/md1 level=raid1 num-devices=2 UUID=71dfc474:...
ARRAY /dev/md2 level=raid1 num-devices=2 UUID=e0f65ea0:...
```

Status

Current information about the RAID status is provided by the /proc/mdstat file. In the following example, there are three RAID-1 arrays, each consisting of two partitions. All three arrays are active and running without errors. Here, [UU] means that the first and the second partition of the array are *up* (i.e., working without problems):

```
root# cat /proc/mdstat
Personalities : [raid0] [raid1] [linear] [multipath]
                [raid6] [raid5] [raid4] [raid10]

md0 : active raid1 sda1[0] sdb1
      979840 blocks [2/2] [UU]
md1 : active raid1 sda2[0] sdb2
      1951808 blocks [2/2] [UU]
```

```
md2 : active raid1 sda3[0] sdb3
      387730624 blocks [2/2] [UU]
unused devices: <none>
```

Now of course, it isn't useful to constantly check this file to see if everything is okay. It's much more convenient to have `mdadm --monitor` take over this task. Usually, this command is started by the init system when the computer boots. However, depending on the distribution, you may need to configure `mdadm` accordingly beforehand. To do this, you want to run the following command on Ubuntu, for example:

```
root# dpkg-reconfigure mdadm
```

Four dialogs then guide you through the `mdadm` configuration. In the first step, you can enable an automatic redundancy check that compares the data on the RAID partitions once a month. This check helps to detect hard disk defects even in areas or files that have not been read or modified for a long time. Internally, the redundancy check is performed by the `checkarray` command, which is started by the `/etc/cron.d/mdadm` Cron script.

In the second and third steps, you activate the monitoring of the RAID status and specify the email address to which any warnings or error reports are to be sent. Internally the monitoring is done by `mdadm --monitor`. On Debian and Ubuntu, the command is started by the init system if `/etc/default/mdadm` contains the `START_DAEMON=true` setting. The email address is stored in `mdadm.conf`. If a problem occurs, `mdadm` sends a notification email to `root`. For this to work, a mail server must be installed on the computer (see Chapter 26)! You can set the email address in `mdadm.conf` using the `MAILADDR` variable.

Finally, in the fourth step, you specify whether your server should start on restart even if it detects a defect in a RAID partition. This is recommended for root servers.

18.15.2 Administration

You usually set up RAID during the installation process (see Chapter 22, Section 22.1). Of course, as in the following examples, you can also create a RAID array at a later stage, set up a file system in it, and integrate it into the directory tree. However, it is impossible (with simple means) to convert an existing root file system to RAID. Thus, the root file system remains the weak point of the overall system.

Setting up a RAID-1 array

The starting point for the following example is a virtual machine with three disks. The first disk contains the distribution, while the other two are supposed to be connected to a RAID-1 device:

```
root# lsblk
...
vdb       252:16   0   20G  0 disk
vdc       252:32   0   20G  0 disk
```

For this purpose, you can use `parted` to set up a partition each on `/dev/vdb` and `/deb/vdc` and mark them for RAID use. The following commands apply to `/dev/vdb`:

```
root# parted /dev/vdb
(parted) mklabel gpt
(parted) mkpart part1 1mib -1mib
(parted) set 1 raid on
(parted) unit mib
(parted) print
Number  Start    End       Size        Name     File system     Flags
 1      1.00MiB  20479MiB  20478MiB    part1                    raid
(parted) quit
```

A single `mdadm` command is sufficient to create a RAID-1 array from the `/dev/vdb1` and `/dev/vdc1` partitions:

```
root# mdadm --create /dev/md0 --level=0 --raid-devices=2 \
  /dev/vdb1 /dev/vdc1
mdadm: Defaulting to version 1.2 metadata
mdadm: array /dev/md0 started.
```

Next, you need to create a file system on the new virtual `/dev/md0` partition. This partition can be mounted into the Linux file system via `mount`. Here, the partition is addressed via the `/mnt/safedata` directory; of course, you can also use another name instead:

```
root# mkfs.ext4 /dev/md0
root# mkdir /mnt/safedata
root# mount /dev/md0 /mnt/safedata
```

If everything works, you should add the new partition to `/etc/fstab`. In all current Linux distributions, the RAID system is automatically initialized by the init system at the next system start:

```
# in /etc/fstab
/dev/md0    /mnt/safedata   ext4    defaults    0 0
```

You must also add a line to the `mdadm.conf` configuration file that describes the new RAID-0 array. `mdadm --examine --scan` provides the line in the required syntax:

```
root# mdadm --examine --scan >> /etc/mdadm.conf
```

Testing a RAID-1 array

To test the functionality of a RAID-1 array—preferably before you have stored critical data there—you should flag a partition as *defective*:

```
root# mdadm /dev/md0 --fail /dev/vdc1
```

Provided that `mdadm --monitor` was started as part of the system startup, `root` on the local machine should now immediately receive an email notification. Otherwise, you can continue to use the array; however, all changes are now stored only on the remaining disk partition. `/proc/mdstat` now shows the `_U` status. This means that one partition is running (`U` for *up*) and one is missing (`_`):

```
root# cat /proc/mdstat
md0 : active raid1 vdb1
      979840 blocks [2/1] [_U]
```

To add `/dev/vdc1` back to `/dev/md0`, you must first explicitly remove the partition that is flagged as defective:

```
root# mdadm --remove /dev/md0 /dev/vdc1
root# mdadm --add    /dev/md0 /dev/vdc1
```

The automatic resynchronization of the two partitions then begins, which takes quite some time depending on the size of the network (approximately 20 minutes per 100 GiB for conventional hard disks; significantly less for SSDs). You can continue to work during this time, but the file system will respond more slowly than usual:

```
root# cat /proc/mdstat
md0 : active raid1 vdb1 vdc1[2]
      485454656 blocks [2/1] [U_]
      [>.....................]  recovery =  3.0% (14577856/485454656)
                               finish=72.8min speed=107724K/sec

root# mdadm --detail /dev/md0   (while synchronization is running)
...
          State : clean, degraded, recovering
 Active Devices : 1
Working Devices : 2
 Failed Devices : 0
  Spare Devices : 1
 Rebuild Status : 75% complete
...
```

```
    Number   Major   Minor   RaidDevice State
       0       3       3        0      active sync        /dev/vdb1
       1       0       0        -      removed
       2      22       2        1      spare rebuilding   /dev/vdc1

root# mdadm --detail /dev/md0   (after completion of synchronization)
...
          State : clean
 Active Devices : 2
Working Devices : 2
 Failed Devices : 0
  Spare Devices : 0
...
    Number   Major   Minor   RaidDevice State
       0       3       3        0      active sync   /dev/vdb1
       1      22       2        1      active sync   /dev/vdc1
```

Disabling a RAID array

`mdadm --stop` disables a RAID array. The file system contained by it must first be removed from the directory tree by using `umount`:

```
root# umount /mount-directory/
root# mdadm --stop /dev/md0
```

Reenabling the RAID array

Only if you have not made any changes to the underlying partitions after `mdadm --stop` can you reassemble and activate the RAID array via `mdadm --assemble` without any data loss:

```
root# mdadm --assemble /dev/md0 /dev/vdb1 /dev/vdc1
mdadm: /dev/md0 has been started with 2 drives.
```

Analyzing partitions

Context information (metadata) has been saved in a special block in all hard disk partitions that you have assembled into RAID partitions using `mdadm`. You can read this information using `mdadm --query`—for example, to determine the status of an unknown system:

```
root# mdadm --query /dev/sda3
/dev/vdb1: is not an md array
/dev/vdb1: device 0 in 2 device active raid1 /dev/md0.
Use mdadm --examine for more detail.
root# mdadm --query /dev/md0
/dev/md0: 19.98GiB raid1 2 devices, 0 spares.
Use mdadm --detail for more detail.
```

`mdadm --examine` provides detailed information about a partition that is part of a RAID array:

```
root# mdadm --examine /dev/sda3
/dev/vdb1:
     Raid Level : raid1
   Raid Devices : 2
 Avail Dev Size : 41904129 (19.98 GiB 21.45 GB)
     Array Size : 20952064 (19.98 GiB 21.45 GB)
            ...
    Device Role : Active device 0
    Array State : AA ('A' == active, '.' == missing, 'R' == replacing)
```

Similarly, `mdadm --detail` provides detailed information about a RAID array:

```
root# mdadm --detail /dev/md0
/dev/md0:
        Raid Level : raid1
              ...
    Active Devices : 2
   Working Devices : 2
    Failed Devices : 0
     Spare Devices : 0
              ...
    Number   Major   Minor   RaidDevice State
       0     252       17        0      active sync   /dev/vdb1
       1     252       33        1      active sync   /dev/vdc1
```

Checking the file integrity

How do you know that all redundantly stored data is really correct? Usually, the RAID system performs integrity tests only when it reads or writes files. However, many files are often not touched for months. Thus, in order to determine with certainty that the hard disks are OK, the RAID system must read all the data blocks and compare the redundant data. This process is also referred to as *scrubbing*:

```
root# echo check > /sys/block/mdn/md<n>/sync_action
```

If errors occur in the process, they can be repaired:

```
root# echo repair > /sys/block/mdn/md<n>/sync_action
```

On Debian and Ubuntu, this is taken care of by the `/usr/share/mdadm/checkarray` script, which is started monthly via `cron`. On Fedora, the same function is performed by the `/etc/cron.d/raid-check cron` script.

Deleting RAID metadata

Storing RAID metadata in otherwise unused sectors of the RAID partition is usually a useful thing to do. However, if you want to use the hard disk differently at a later time, the RAID metadata can turn into a problem: Linux installation programs and `mdadm` recognize the remnants of the RAID configuration and refuse to see that these partitions should now be used differently. To solve this problem, run the following command, which must be applied to all RAID partitions:

```
root# mdadm --zero-superblock /dev/vdb1
```

18.15.3 Replacing a Defective RAID-1 Hard Disk

The starting point for this section is a Linux distribution where a RAID-1 setup has already been set up during the installation process (see Chapter 22, Section 22.1). The device names for this example are `sda` for the first disk and `sdb` for the second (defective) disk:

```
/dev/sda1:              Partition without RAID
/dev/sdb1:              Partition without RAID
/dev/sda2 + /dev/sdb2: RAID-1  ->  /dev/md0
/dev/sda3 + /dev/sdb3: RAID-1  ->  /dev/md1
```

If the defective hard disk or SSD is still active in the RAID mounts, you must explicitly remove all partitions of this hard disk:

```
root# mdadm --remove  /dev/md0 /dev/sdb2
root# mdadm --remove  /dev/md1 /dev/sdb3
```

Installing a replacement disk

After that, you need to get a replacement hard disk as soon as possible. The new hard disk must have enough space to create partitions on it that are as large as those on the existing hard disks.

Make sure that you really do remove the defective hard disk and not the still working one by mistake! This recommendation sounds trivial, but when a computer contains two or more identical hard disks, it isn't that easy to find the right hard disk. Only the serial number is unique! You can find out which serial number is associated with which device name via `hdparm` or `smartctl`. Both commands can only be run if the package of the same name has been installed upfront:

```
root# smartctl -i /dev/sdb
...

Device Model:     SAMSUNG HD403LJ
Serial Number:    SONFJ1MPA07356
```

```
root# hdparm -i /dev/sdb
/dev/sdb:
...

Model=SAMSUNG HD403LJ, FwRev=CT100-12,
SerialNo=SONFJ1MPA07356
```

Cloning partitions

After replacing the hard disk, you must create new partitions on the new hard disk that are at least as large as the existing RAID partitions. The partitions must also be tagged as RAID partitions. You can avoid this work by using `sfdisk` to duplicate or clone (`sdb`) the MBR partition table from one hard disk (here `sda`) to a second one:

```
root# sfdisk -d /dev/sda | sfdisk /dev/sdb
```

If GPT is involved, two `sgdisk` commands will produce the desired result. The first command duplicates the partitions, while the second one assigns new random identification numbers (GUIDs) to them. You may need to install the `gdisk` package beforehand:

```
root# sgdisk /dev/sda -R /dev/sdb
root# sgdisk -G /dev/sdb
```

Restoring the RAID array

After this preparatory work, the rest is a piece of cake. You need to add the partitions of the new hard disk to the RAID mounts:

```
root# mdadm --add /dev/md0 /dev/sdb2
root# mdadm --add /dev/md0 /dev/sdb3
```

Then the kernel starts to synchronize the partitions of the new hard disk with the existing RAID data. You can track the status of the synchronization via `cat /proc/mdstat`.

GRUB reinstallation into the MBR

On a BIOS computer, GRUB must usually be installed in the MBR of the replaced hard disk (here `sdb`):

```
root# grub-install  /dev/sdb     (Debian, Ubuntu)
root# grub2-install /dev/sdb     (Fedora, RHEL)
```

Restoring an EFI partition

If, on the other hand, you have provided an EFI partition on each disk on an EFI computer, you must restore this partition on the replaced disk. The required commands may look like the ones ahead, for example (see Chapter 22, Section 22.1). You must of course adapt the device names and mount directories to your own environment:

```
root# mkfs.vfat /dev/sdb1
root# mount /dev/sdb1 /boot/alt-efi/
root# rsync -a /boot/efi/ /boot/alt-efi
root# blkid /dev/sdb1
/dev/sdb1: UUID="0D35-4FE1" TYPE="vfat" PARTUUID="929714e2-...24b2"
```

Then you need to enter the new UUID of the EFI partition in `/etc/fstab`:

```
# file /etc/fstab
...
UUID=0D35-4FE1   /boot/efi  vfat nofail,umask=0077  0  1
```

Finally, you must inform EFI that it must consider the recovered EFI partition in the boot process in the future:

```
root# efibootmgr  --create --disk /dev/sdb --part 1 \
                  --label 'ESP on sdb' --loader '\EFI\ubuntu\
shimx64.efi'
```

Practicing for an Emergency

I strongly recommend that you try out a RAID repair on a test system at your leisure!

You can simulate a hard disk defect by marking a partition as defective using `mdadm--fail` or by temporarily disconnecting the cable to a hard disk (but of course not during operation!).

It's much easier to try out RAID in virtual machines. There, you can simply create a new virtual machine with two, three, four, or even more virtual disks.

18.16 Logical Volume Manager

The LVM puts a logical layer between the file system and the partitions of the hard disk. The principle, merits, and terminology of LVM have already been explained in Chapter 2, Section 2.6. This section focuses on administrating LVM.

LVM glossary

Before you can use LVM, you must create a data pool, called a *volume group* (VG). *Physical volumes* (PVs) serve as the building blocks of a volume group. These are specially marked partitions or RAID devices.

Within the pool you can define *logical volumes* (LVs). These are areas composed of memory blocks that later hold a file system. These memory blocks must always be a multiple of a *physical extent* (PE; 4 MiB by default).

dm_mod Inside Linux, the `dm_mod` kernel module that you already know from RAID is responsible for LVM. In some distributions, the LVM functions are compiled directly into the kernel and therefore do not appear in the `lsmod` result.

RAID You can combine LVM and RAID. To do this, you must first set up a RAID array and then use the resulting `/dev/mdN` device as the PV.

LVM commands There is a whole range of commands available for administrating LVM. The names of the commands start with `pv`, `vg`, or `lv`, depending on whether they are intended for editing physical volumes, volume groups, or logical volumes. The most important representatives are listed in Table 18.8.

Command	Function
`lvcreate`	Sets up a new LV in a VG
`lvdisplay`	Provides detailed information about a LV
`lvextend`	Enlarges a LV
`lvreduce`	Reduces the size of a LV
`lvremove`	Deletes a LV
`lvrename`	Assigns a new name to the LV
`lvscan`	Lists all LVs
`pvcreate`	Marks a partition or device as a PV
`pvdisplay`	Provides detailed information about a PV
`pvremove`	Removes the PV identification of an unused PV
`pvscan`	Lists all PVs
`vgchange`	Changes the attributes of a VG
`vgcreate`	Creates a new VG from one or more PVs
`vgdisplay`	Provides detailed information about a VG
`vgextend`	Increases a VG by one PV
`vgmerge`	Combines two VGs
`vgreduce`	Reduces a VG by one unused PV
`vgrename`	Assigns a new name to a VG
`vgscan`	Lists all VGs

Table 18.8 Overview of LVM Commands

The commands are part of the lvm2 package, which may need to be installed first. Instead of the individual commands, you can also execute the entire LVM administration using the lvm command, passing the desired command as the first parameter. Thus, the lvcreate and lvm lvcreate commands are equivalent.

Examples

The following examples show the application of some LVM commands. Here I assume that no LVM was set up during the installation. Now the additional hard disk, /dev/sdc, is to be used via LVM. To try it out, you should use parted to set up two partitions (/dev/sdc1 and /dev/sdc2 in the following examples) and mark the partitions for use by LVM by using set <n> lvm on.

To initialize LVM, run modprobe and vgscan. Once an LVM system has been set up, the LVM kernel module is automatically executed during computer startup. Thus, the manual initialization is required only the first time:

```
root# modprobe dm_mod
root# vgscan
  Reading all physical volumes (this may take a while...)
  No volume groups found
```

For didactic reasons, I first want to set up LVM on the /dev/sdc1 partition and later extend the LVM system with /dev/sdc2. If it's already clear that you want to use the entire disk for LVM, it is of course easier to mark a partition of the maximum size using pvcreate for LVM use:

```
root# pvcreate /dev/sdc1
  Physical volume "/dev/sdc1" successfully created
```

Now all PVs must be combined into one VG. In this example, there is only one PV for now, but the step is still necessary. The name you want to assign to the VG must also be passed to the vgcreate command. In this example, the VG is given the name myvg1:

```
root# vgcreate myvg1 /dev/sdc1
  Volume group "myvg1" successfully created
```

myvg1 now represents a kind of data pool, which is still unused at this point. To use it, you need to set up an LV inside myvg1, which is a kind of virtual partition. To do this, you must pass three pieces of information to the lvcreate command—the required size of the LV, the name of the new LV, and the name of the existing VG:

```
root# lvcreate -L 2G -n myvol1 myvg1
  Logical volume "myvol1" created
```

The command also creates the /dev/myvg1/myvol1 file, which is actually a link to the /dev/mapper/myvg1-myvol1 file. The LV can now be used under one of these two device names like an ordinary hard disk partition. For example, to set up a file system in a logical volume, you can use mkfs.ext4 or mkfs.xfs:

```
root# mkfs.ext4 /dev/myvg1/myvol1
```

By using mount, you can make sure that everything worked correctly:

```
root# mkdir /test
root# mount /dev/myvg1/myvol1 /test
```

Enlarging the file system

One reason for using LVM at all is to be able to enlarge a file system at a later stage without having to repartition the hard disk. In the following example, the file system that was set up earlier (/dev/myvg1/myvol1 via /test) will be enlarged from the original 2 GiB to 3 GiB. df shows the capacity of /test before the change:

```
root# df -h -T /test
Filesystem   Type    Size Used  Avail Used% Mounted on
/dev/mapper/myvg1-myvol1
             ext4    2.0G  76OM  1.2G  40% /test
```

For this purpose, you must first increase the size of the logical volume. To do this, you need to pass the device name and the new size to lvextend. After that, the ext4 file system will also be enlarged accordingly:

```
root# lvextend -L 3G /dev/myvg1/myvol1
  Extending logical volume myvol1 to 3.00 GB
  Logical volume myvol1 successfully resized
root# resize2fs /dev/myvg1/myvol1
```

df proves that everything worked:

```
root# df -h -T /test
Filesystem   Type    Size Used  Avail Used% Mounted on
/dev/mapper/myvg1-myvol1
             ext4    3.0G  76OM  2.1G  27% /test
```

In general, a reduction in size is also possible. However, to do this, you must first remove the relevant file system from the directory tree, check it using fsck.ext4, and finally resize it using resize2fs. Only at this point can you use lvreduce to downsize the underlying LV.

As long as there is still space in the storage pool (in the volume group), logical volumes can easily be enlarged. But what do you do when the VG is also full?

In that case, you must create a new partition on any hard disk of your computer, set up this partition as a physical volume, and add it to the volume group using `vgextend`. The following two commands demonstrate this for the `/dev/sdc2` partition:

```
root# pvcreate /dev/sdc2
  Physical volume "/dev/sdc2" successfully created
root# vgextend  myvg1 /dev/sdc2
  Volume group "myvg1" successfully extended
root# vgdisplay myvg1
...
  VG Size               18.64 GB
  Alloc PE / Size       640 / 2.50 GB
  Free  PE / Size       4132 / 16.14 GB
...
```

Snapshots

LVM enables you to create snapshots. A *snapshot* is an unalterable image of the file system at a particular point in time. The snapshot can be integrated into the directory tree like a separate file system. If the underlying file system changes, the original data for the snapshot will be archived. You must specify how much space LVM should reserve for this purpose when you create the snapshot. If this space is used up, the snapshot becomes invalid and can no longer be used.

LVM snapshots provide much less functionality than `btrfs` snapshots. LVM snapshots are usually used for backups. They ensure that the files do not change during the backup, so the backup is consistent.

The following commands show how you can first create a snapshot of the `myvol1` LV, mount it into the directory tree in the `/media/backup` directory, create a backup of it, remove the snapshot from the directory tree again, and finally delete it.

While the backup is running, the `myvol1` LV can continue to be used without any restriction (e.g., as a storage space for a database server). However, the changes implemented during the backup must not exceed 100 MiB. During the backup, you can use `lvdisplay /dev/vg1/snap` to determine what percentage of this space is already in use:

```
root# lvcreate -s -L 100M snap /dev/myvg1/myvol1
root# mkdir /media/backup
root# mount /dev/vg1/snap /media/backup
```

```
root# backup-script /media/backup    (create backup)
root# umount /media/backup
root# lvremove /dev/vg1/snap
```

Note that you must specify the device name of the underlying logical volume via `/dev/vgname/lvname`, not via `/dev/mapper/vgname-lvname`!

18.17 Self-Monitoring, Analysis, and Reporting Technology

SMART is a feature of almost all commercially available hard disks and SSDs. Thanks to SMART, various parameters of the data carrier are saved regularly. These parameters allow you to draw conclusions about possible defects of the hard disk and its expected lifetime.

The SMART parameters can be read via a special interface. Regular monitoring of the parameters by the operating system is a kind of early warning system. This section provides an overview of the tools available on Linux for reading the SMART parameters.

Dream and Reality

Ideally, thanks to SMART, you will be informed in good time before a disk fails. However, studies have shown that this rarely works in practice, neither with conventional hard disks nor with SSDs! That doesn't mean you shouldn't use SMART—but by no means should you neglect your backups just because SMART thinks your disks are still in good shape.

smartctl

The `smartctl` command determines the SMART status. This is part of the `smartmontools` package in most distributions. Depending on the distribution, this also automatically installs an email server (MTA) to send SMART notifications via email. This is useful on a server, but usually not on a desktop computer. For Debian and Ubuntu, you can avoid installing the email server by passing the `--no-install-recommends` option to `apt`.

In its simplest implementation, `smartctl` provides various status information. If `smartctl -i` reports *SMART support is: Disabled* in the last line, you can enable SMART by using `smartctl -s on`. Note that `smartctl` often doesn't work with external hard drives (USB). The following data comes from a server hard disk:

```
root# smartctl -i /dev/sdb
Model Family:     Seagate Enterprise Capacity 3.5 HDD
...
```

```
User Capacity:    4,000,787,030,016 bytes [4.00 TB]
Sector Sizes:     512 bytes logical, 4096 bytes physical
Rotation Rate:    7200 rpm
Form Factor:      3.5 inches
Device is:        In smartctl database [for details use: -P show]
...
SMART support is: Available - device has SMART capability.
SMART support is: Enabled
```

`smartctl -H` or `smartctl --health` indicates whether the hard disk is currently in good condition and will probably continue to function for the next 24 hours:

```
root# smartctl -H /dev/sdb
...

SMART overall-health self-assessment test result: PASSED
```

Condition of a conventional hard disk

If `smartctl` does not return *PASSED* as a result here, you should *immediately* start performing a full backup!

`smartctl -A` or `smartctl --attributes` returns a list of vendor-specific hard disk attributes. There is no fixed standard for these attributes, but the most important attributes are supported by many hard disk manufacturers. When you interpret the values, two columns are crucial: `VALUE` indicates the current value and `THRESH` the threshold value. If the current value falls below the threshold value or approaches 0 (for `THRESH=0`), problems are to be expected or the hard disk has reached its intended service life.

Most values are normalized to a base value of 100. For example, `Power_On_Hour` starts with the value 100 for a new hard disk. After a certain number of operating hours, the value drops to 99, and so on.

The `TYPE` column indicates the importance of the value for the "health" of the disk. Critical lines are marked with `Pre-fail`, less important ones with `Old_age`. (Who came up with that?) If `VALUE` approaches 0 in a `Pre-fail` line or is already close to the `THRESH` limit, the hard disk must be replaced urgently.

The following abbreviated results come from the same server hard disk as in the previous example:

```
root# smartctl -A /dev/sdb
...
smartctl 6.6
Vendor Specific SMART Attributes with Thresholds:
```

```
ID# ATTRIBUTE_NAME          VALUE WORST THRESH TYPE      RAW_VALUE
  1 Raw_Read_Error_Rate     078   064   044    Pre-fail  56345339
  3 Spin_Up_Time            094   094   000    Pre-fail  0
  4 Start_Stop_Count        100   100   020    Old_age   19
  5 Reallocated_Sector_Ct   100   100   010    Pre-fail  136
  7 Seek_Error_Rate         095   060   045    Pre-fail  2990863242
  9 Power_On_Hours          035   035   000    Old_age   57104
 10 Spin_Retry_Count        100   100   097    Pre-fail  0
 12 Power_Cycle_Count       100   100   020    Old_age   18
184 End-to-End_Error        100   100   099    Old_age   0
...
194 Temperature_Celsius     036   046   000    Old_age   36 (...)
...
198 Offline_Uncorrectable   100   100   000    Old_age   0
199 UDMA_CRC_Error_Count    200   200   000    Old_age   0
```

The following notes should help you better understand the data:

- The `ID=1` line indicates how often correctable read errors occur. The current value is 78, but in the past the value has also dropped to 64. The limit is 44, so fortunately still far away.
- According to line `ID=5`, it has only very rarely been necessary to replace broken sectors with reallocated sectors. The start value 100 is still displayed. That's a good sign.
- In line `ID=9`, `RAW_VALUE` indicates the operating hours. If you divide this number by 365 days and 24 hours, the operating time is 6.5 years! The `VALUE` column contains 35. This is already quite far from the starting value of 100, but the remaining life is still around 35%! So, this hard disk is intended for long use. Interestingly, this line is marked with `TYPE=Old_age`. For the functioning of the hard disk, the operating hours are less critical than various other parameters.
- The `ID=194` line indicates the current temperature of the hard disk: 36 degrees Celsius is not a problem. The usual scaling from 100 (good) to 0 (bad) does not apply to this line! The `WORST` column indicates how hot the hard disk has become at maximum in the past; 46 degrees Celsius is not a cause for concern.
- Line `ID=199` is remarkable in that the values of this line do not start at 100, but at 200. No UDMA CRC errors have been detected so far.

Here's a brief summary: despite its considerable age, the hard disk seems to be in surprisingly good condition. However, if this was my server, I would still plan to replace the hard disk (or better yet, the entire server) soon.

Fault-free continuous operation for more than six years must actually already be considered a stroke of luck.

Further Reading

Much more detail on interpreting SMART data is summarized at the following great website: *https://wiki.unraid.net/Understanding_SMART_Reports*.

`smartctl -l error` provides information about the five most recent errors. Often the result is simply empty (*No Errors Logged*). As a rule, isolated errors that do not recur are no cause for alarm:

```
root# smartctl -l error /dev/sda
SMART Error Log Version: 1
No Errors Logged
```

Performing a self-test

SMART provides for different variants of self-tests to determine the current state of the hard disk even more precisely. You can start such tests using `smartctl -t short/long`. A short test takes a few minutes, while a detailed test (`long`) may take several hours. The test is performed in the background so you can continue working normally.

Once the end of the test has been reached, you can view the result by running `smartctl -l selftest`. The *Remaining* column indicates the extent to which the self-test has already been performed. If the value is greater than 0%, the test is still running! *LifeTime* indicates how many hours the hard disk has already been in operation.

LBA indicates the location (sector) of the first error. In the following result, nine self-tests were performed—one of them immediately after installation of the panel after six hours of operation, the others at irregular intervals:

```
root# smartctl -t short /dev/sda
root# smartctl -l selftest /dev/sda
Num Test_Description  Status          Remaining  LifeTime(hours)  LBA
# 1 Short offline     Completed without error       00%   57104    -
# 2  Extended offline Completed without error       00%   18061    -
...
# 8  Extended offline Completed without error       00%      25    -
# 9  Extended offline Completed without error       00%       6    -
```

Status of NVMe SSDs

If you apply `smartctl` to modern NVMe SSDs, you will get results that look quite different from conventional hard disks. This is because the SMART functions for NVMe disks have been reimplemented internally. Many functions, such as those for performing self-tests, are missing altogether.

The following listing shows the results of a modern NVMe SSD (Samsung PM981) after about 2.5 years of continuous use in a server. What is confusing about it is that the `Power On Hours` line provides a result that does not match this at all. According to this, the disc would only be in operation for just under 10,000 hours (a little over a year). On the internet, I have found several reports that the operating hours often do not correspond to reality. The lifespans of SSDs are primarily limited by the amount of data stored. Unfortunately, I did not find this information in any data sheet for the disk used here, but Samsung specifies a limit of 300 terabytes written (TBW) for comparable models with a capacity of 512 GB. `smartctl` reveals that only 20 TB has been written for the disk at hand. Furthermore, the `Percentage Used` line indicates that the disk is still in almost new condition:

```
root# smartctl -i /dev/nvme0n1
Model Number:                       SAMSUNG MZVLB512HBJQ-00000
Total NVM Capacity:                 512,110,190,592 [512 GB]
root# smartctl -A /dev/nvme0n1
Critical Warning:                   0x00
Temperature:                        38 Celsius
Available Spare:                    100%
Available Spare Threshold:          10%
Percentage Used:                    2%
Data Units Read:                    95,198,942 [48.7 TB]
Data Units Written:                 39,679,943 [20.3 TB]
Host Read Commands:                 855,328,520
Host Write Commands:                1,486,041,469
Controller Busy Time:               12,979
Power Cycles:                       7
Power On Hours:                     9,779
...

Temperature Sensor 1:               38 Celsius
Temperature Sensor 2:               40 Celsius
```

Automatic monitoring (smartd)

`smartctl` is certainly an interesting tool to collect information about the hard disk. However, the command is too unwieldy for regular monitoring of all hard disks. This task is performed by the `smartd` program, which is actually a daemon (system service). The commands for automatic startup vary depending on the distribution and are summarized in Chapter 10, Section 10.5.2. `smartd` is controlled by `/etc/smartd.conf`. In some distributions, the init script also analyzes the `/etc/sysconfig/smartmontools` or `/etc/default/smartmontools` files. These files contain additional command options for `smartd`. For Debian and Ubuntu, you need to set `start_smartd=yes` in `smartmontools`!

A simple configuration for a computer with two SATA hard disks (/dev/sda and /dev/sdb) looks as follows:

```
# file /etc/smartd.conf
/dev/sda -d sat -H -m root -M test
/dev/sdb -d sat -H -m root -M test
```

This means that the "health" of the specified disks is monitored every half hour (as by `smartctl -H`). If an error is detected, `smartd` sends an email to the local `root` user. (However, this requires a local email server.) `-d sat` identifies the hard disks as SATA devices. `-M test` is used to test whether the email dispatch works. Start `smartd` as follows:

```
root# systemctl start smartmontools
```

If you receive the test email, you can remove `-M test` from the configuration. A lot more configuration examples can be found in the `smartd.conf` file that comes with `smartmontools`.

18.18 SSD TRIM

SSDs differ from conventional hard disks in one respect: For the internal optimization of the memory cells, it's necessary that the operating system informs the SSD about which memory blocks of the file system are currently not in use (e.g., because a file has been deleted). Like the underlying command, this process is called *TRIM*. You can read about its technical background at Wikipedia (*https://en.wikipedia.org/wiki/Trim_(computing)*).

Batch TRIM

Current Linux distributions run a so-called batch TRIM once a week using a systemd timer. All the blocks that have become free in the past week are thus cleaned up at once. For most installations, this is a good solution:

```
root#   systemctl list-timers fstrim
NEXT            LAST            UNIT          ACTIVATES
Mon 2023-05-15  Mon 2023-05-08  fstrim.timer  fstrim.service
```

Online TRIM

An alternative approach is an online TRIM, where Linux immediately notifies the SSD whenever a file is deleted. However, the disadvantage of this method is that the SSD-internal cleanup tasks are performed just when the SSD is busy anyway—namely, during intensive write operations. Thus, online TRIM slows down *every* write operation. The good thing about it is that the batch TRIM can be omitted.

To enable the online TRIM procedure, you must add the `discard` option to the `/etc/fstab` file for the relevant file systems. This option is available for the `ext4`, `btrfs`, and `xfs` file systems. Make sure not to include syntax errors in `/etc/fstab`! The list of `mount` options must not contain any blank spaces:

```
# file /etc/fstab
UUID=018e... / ext4 errors=remount-ro,user_xattr,discard 0 1
```

Manual batch TRIM

You can also decide against any automation and only trigger a manual batch TRIM occasionally. To do this, you need to run `fstrim` in a terminal:

```
root# fstrim -v /
/: 95.4 GiB (102407581696 bytes) trimmed
```

For encrypted file systems, the command may return the following error message: *the discard operation is not supported*. In this case, you must explicitly include the `discard` option in `/etc/crypttab`. However, this will reduce the encryption security.

You can also limit the TRIM command to larger data blocks, which will only report free data blocks of the specified minimum size to the SSD. `fstrim` can then be executed much faster and still achieves a very good effect:

```
root# fstrim -m 64K -v /
```

18.19 Encryption

Notebook computers and USB flash drives are often lost or stolen. What is often worse than the actual loss of the device is the fact that important data falls into the wrong hands: private files, SSH keys, company secrets, and so on. This is unnecessary. A relatively simple encryption of the file system is sufficient to effectively protect the data.

This section provides some background information on dealing with encrypted files and file systems.

As a rule, you can set up an encrypted file system when you install Linux. Encryption of already existing file systems at a later stage is not possible. (You can, however, encrypt new file systems, of course. I will present the required commands ahead.)

18.19.1 Encrypt Individual Files

gpg

The easiest way to encrypt a single file is to use the `gpg` command. `gpg -c` prompts you twice for a password, then encrypts the specified file and

saves the result under the name file.gpg. The CAST5 encryption algorithm is used by default. You can then delete the original file. gpg -d restores the file:

```
user$ gpg -c file
Enter passphrase: **********
Please re-enter this passphrase: **********
user$ gpg -d file.gpg > file
Enter passphrase: **********
```

gpg can also use a public or private key for encoding or decoding, sign files, manage keys, and more. Accordingly, the description of the countless options on the man page is around 50 pages long. However, the manual use of gpg is rather uncommon. More commonly, gpg is used by email clients to sign or encrypt emails.

18.19.2 Encrypting a File System

dm_crypt and LUKS

Countless methods for encrypting file systems have been developed in the past, including CryptoFS, eCryptfs, Enc-FS, Loop-AES, and LUKS. Some of these methods are still in use while others have been discarded—often for safety reasons. *Linux Unified Key Setup* (LUKS) version 2 is currently the most popular version. LUKS is based on the dm_crypt kernel module, which adds cryptography functions to the Linux device mapper also used for LVM. The module is a logical layer between the encrypted raw data on the hard disk and the file system as the Linux user sees it. dm_crypt supports various encryption algorithms. dm_crypt is often combined with LVM, but this is by no means necessary. You can also use dm_crypt on a system without LVM.

LUKS adds a header with meta information to the encrypted data. Among other things, the header specifies which method is used to encrypt the data. LUKS simplifies the integration of encrypted volumes in Linux quite considerably.

cryptsetup

To set up encrypted file systems, you need to use the cryptsetup command from the namesake package. The following lines show how you can format a USB flash drive (/dev/sdh1) first as a crypto device (luksFormat) and then activate the device under the arbitrarily chosen name mycontainer (luksOpen). By its very nature, your data is only as secure as your password or passphrase, which consists of several words. The recommended password length is as long as possible. After that you can use /dev/mapper/mycontainer like a hard disk partition or LV—that is, set up a file system, include it in the directory tree, and so on. After using umount, you must remember to disable

the crypto device again (luksClose) in order to release /dev/sdh1. Only at this point can you remove the USB flash drive:

```
root# cryptsetup luksFormat --type luks2 /dev/sdh1
Data on /dev/sdh1 will be irrevocably overwritten.

Are you sure? (Type uppercase yes): YES
Enter LUKS passphrase: **********
Verify passphrase: **********
Command successful.
root# cryptsetup luksOpen /dev/sdh1 mycontainer
Enter LUKS passphrase: **********
root# mkfs.ext4 /dev/mapper/mycontainer
root# mount /dev/mapper/mycontainer /test
root# touch /test/xy
root# umount /test/
root# cryptsetup luksClose mycontainer
```

Of course, you can also use a partition of an internal or external hard disk, a RAID device, or a logical volume of your LVM system instead of a USB flash drive. To do this, simply replace /dev/sdh1 with the device name of the partition or LV.

By default, cryptsetup uses the AES encryption algorithm with a key length of 512 bits. You can see this for yourself by running cryptsetup luksDump. This command provides the crypto meta information that LUKS stores in a special sector of the disk:

```
root# cryptsetup luksDump /dev/sdh1
LUKS header information for /dev/sdh1
Version:        2
...

Keyslots:
  0: luks2
        Key:        512 bits
        Priority:   normal
        Cipher:     aes-xts-plain64
        Cipher key: 512 bits
Keyslots:
  0: luks2
    Key:        512 bits
    Priority:   normal
    Cipher:     aes-xts-plain64
```

```
    Cipher key: 512 bits
    PBKDF:      argon2i
...
```

If you want to use a different encryption algorithm or a longer key, you must pass the relevant data to `cryptsetup luksFormat` with the `-c` and `-s` options. `cat /proc/crypto` shows which algorithms are available.

Using `cryptsetup luksAddKey`, you can secure access to a LUKS device with a total of eight different passwords. This allows the shared use of a disk, with each user using their own password.

Security

In April 2023, a case came to light in which the police gained access to a LUKS-encrypted file system. The exact circumstances remained unclear.

It's possible that *Password-Based Key Derivation Functions* (PBKDF; see the last line in the previous listing) used in the past constitute a problem. The best current protection against brute force attacks is `argon2id`. (The LUKS container in the previous listing, on the other hand, uses `argon2i`.) You can find the relevant background information on this topic on the following web page: *https://mjg59.dreamwidth.org/66429.html*.

Apart from this, the following applies: use a password that is as long as possible! This makes guessing impossible in a finite amount of time.

Speed

In general, the default settings of `cryptsetup` make sense. According to the current state of the art, the encryption is sufficiently secure *and* very fast with modern CPUs. An overview of the expected speed is provided via `cryptsetup benchmark`:

```
root# cryptsetup benchmark
# The tests are only approximately accurate because they do not
# access the disk.
#   Algorithm | Key | Encryption | Decryption
          ...
        aes-xts        512b      1124.3 MiB/s      1124.1 MiB/s
    serpent-xts        512b       401.3 MiB/s       408.5 MiB/s
    twofish-xts        512b       223.4 MiB/s       223.0 MiB/s
```

With AES encryption with a 512-bit key, the CPU can therefore encrypt more than one GiB per second. Modern PCIe SSDs can manage significantly higher transfer rates. Nevertheless, the efficiency losses caused by encryption should hardly be noticeable in a typical desktop use. (However, benchmark tests prove that encryption noticeably slows down modern

SSDs! If maximum I/O performance is important to you, you should use an encrypted partition for private data and an unencrypted partition for I/O-heavy applications like database servers.)

I have another tip for performance enthusiasts: Where possible, LUKS utilizes 4K blocks. However, many SSDs only communicate with the outside world in 512-byte blocks (presumably to ensure maximum compatibility with hard disks from the Stone Ages.) If you don't have a pro SSD that uses 4K blocks out of the box, you can try enabling 4K mode on your SSD by using the `nvme format` command before using it for the first time. This is possible with some models (not all!). Another option is to coax `cryptsetup` into using 4K blocks by using `--sector-size 4096`, no matter what block size the SSD reports.

If you are interested in this topic, I have put together a recommended reading list for you:

- *https://wiki.archlinux.org/title/Advanced_Format*
- *https://unix.stackexchange.com/questions/93791*
- *https://unix.stackexchange.com/questions/621923*

Personally, I think that tuning the block size is only worthwhile in exceptional cases. Furthermore, it's difficult to influence LUKS parameters during the installation of Linux. This is most likely to succeed with a manual setup of Arch Linux.

luksFormat

`luksFormat` makes it a little easier to set up an encrypted partition or disk. The command first executes `cryptsetup luksFormat` and then `mkfs.vfat`. If you want a different file system type, you must specify it using `-t`.

> **Tip**
>
> After the command has been executed (but often also when errors have occurred), an active crypto device remains: `/dev/mapper/luksformatN`. Before you can remove the disk or set it up as a crypto device again, you must run `cryptsetup luksClose luksFormatN`!

Desktop usage

When you connect LUKS-formatted external disks to your computer and work on GNOME or KDE, the disks are automatically recognized as crypto devices. A dialog appears in which you must specify the encryption password. Then the disk will be mounted into the file system. The container name for `/dev/mapper` is `luks_crypto_NN`. When you unmount the disk, `luksClose` will be executed as well. Using the encrypted disk couldn't be easier!

crypttab

If you've set up an encrypted file system in a partition of a local disk, you probably want this file system to be mounted in the directory tree when the

computer boots. To automate this process, the `cryptsetup` package contains the necessary init scripts. However, these require that the crypto device be entered in the `/etc/crypttab` file.

The structure of this file is simple: the first column specifies the preferred name for `/dev/mapper`, the second column specifies the device name or UUID of the crypto container, the third column specifies the file from which the key is to be read (such as a USB flash drive) or `none` if the encryption password is entered interactively, and the fourth column contains options.

In the following example, the `/dev/sda7` device is to be set up under the name `/dev/mapper/cdisk1`. The password is to be entered during computer startup, and the crypto device has been set up using LUKS. A lot of other options are described in `man crypttab`:

```
# file /etc/crypttab
# Mapper Name   Device            Key File   Options
cdisk1          /dev/sda7         none       luks
```

As in `/etc/fstab`, it is recommended to specify the UUID of the crypto container instead of the device name. You can determine this via `blkid`:

```
root# blkid /dev/sda7
/dev/sda7: UUID="075225d9..." BLOCK_SIZE="4096" TYPE="ext4" PARTUUID=
"60a4..."
```

This then results in the following file:

```
# file /etc/crypttab
# Mapper Name   Device            Key File   Options
cdisk1          UUID=075225d9...  none             luks
```

LUKS and TRIM

By default, LUKS is not compatible with TRIM. If the encrypted device is on an SSD and you want to allow TRIM, you need to add the `discard` option in `/etc/crypttab`. After you have recreated the *initrd* file and rebooted the machine, you can use `fstrim`:

```
# Mapper Name   Device          Key File   Options
cdisk1          /dev/sda7       none             luks,discard
```

Note, however, that TRIM helps an attacker identify free blocks on the SSD. This reduces the encryption security. If you do not want to compromise your security, you should avoid `discard`.

/etc/fstab

To ensure not only that the crypto device is activated, but that its file system is also included in the directory tree, you must also add `/etc/fstab`. The

following line causes the file system to be usable through the /media/private-data directory:

```
# file /etc/fstab
...
/dev/mapper/cdisk1    /media/private-data    ext4    defaults    0 0
```

Then restart the computer and test if everything works.

Enlarging the file system

If you find that the encrypted file system is too small, you can enlarge it retroactively. In doing so, you must follow this order:

- Resize the partition or logical volume.
- Resize the crypto device with `cryptsetup resize <luks-device-name>`.
- Resize the file system—for example, with `resize2fs` or `xfs_growfs`.

Disadvantages

The encryption of a partition is associated with disadvantages. First, all file operations are a bit slower than in normal operation. Second, you need to enter the password every time you boot. Not only is this annoying, but it also makes it impossible to reboot the system in your absence. Basically, encrypted file systems are most often used on notebook computers, but not usually or only in exceptional cases on servers.

18.19.3 Encrypting the Entire System

Using the /etc/crypttab file presented in the previous section, it's only a small step from encrypting a local partition (such as /home) to encrypting the entire system, including the system partition. Two details are important in this context. First, GRUB cannot access the encrypted data, which is why a separate, nonencrypted boot partition is absolutely necessary. Second, to access the system partition, it's necessary to enter the encryption password.

The functions required for this must be integrated as scripts in the *initrd* file. (The cryptsetup package contains all the necessary files.) Prior to that, however, it must be clarified who needs the encryption of the entire system in the first place. As a matter of fact, it should be sufficient to encrypt only the private files in /home. However, the system partition can also be revealing for the notebook thief if they are actually interested in the data: /var/cache or /var/tmp can contain remnants of sent emails, printed documents, read PDFs, and so on; /var/log documents who has worked on the computer and when; the swap partition possibly contains swapped out data blocks with security-critical information; and so on. In a nutshell: if you really want to fully protect your data or your privacy on the computer, you will have to encrypt the entire system, for better or worse.

Installation

Most major distributions provide an option in the installer to encrypt the entire system. For Debian and Ubuntu, you can select the **Encrypted LVM System** partitioning variant; for Fedora, you need to enable the **Encrypted System** option in the partitioning dialog. On openSUSE, you want to select the **LVM-based** and **Encrypted** options for partitioning.

System structure

The structure of the encrypted system looks uniform in most distributions: an unencrypted boot partition for GRUB is created, as well as a second partition that is encrypted and serves as a physical volume for LVM. This way, all partitions set up via LVM (swap partition, system partition, data partitions) are automatically encrypted. Furthermore, it isn't necessary to define a separate password for each partition; rather, a central password for the entire LVM system is sufficient. Figure 18.4 illustrates the structure of such a system, whereby I have taken the designation of the devices or LVs from Fedora.

Of course, there are many alternatives to the setup presented in Figure 18.4. One possible variant is to omit LVM and encrypt each partition separately. You can use a random key for the swap partition, which is generated from `/dev/urandom` each time the system boots.

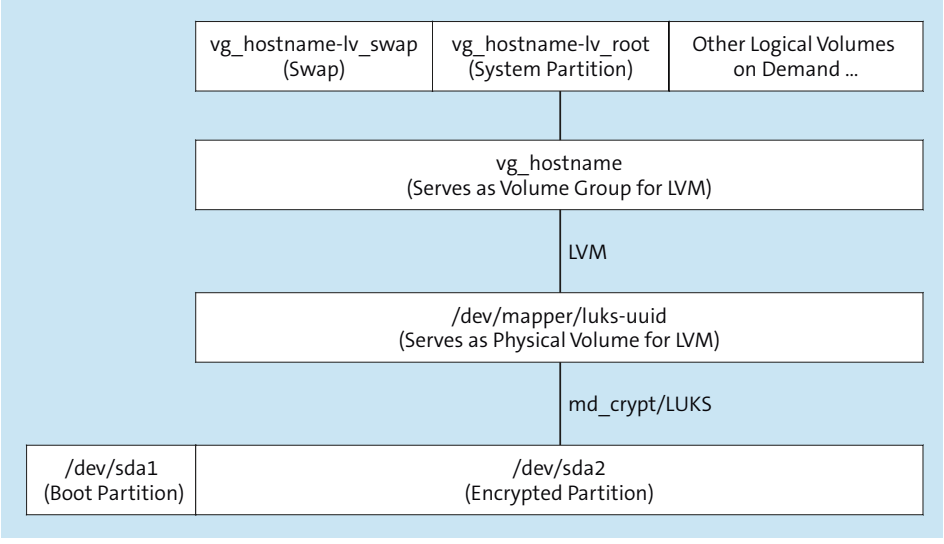


Figure 18.4 Fully Encrypted Linux System

Another way to specify the key is also conceivable: instead of interactively entering a password, the key can be read from a file on a USB flash drive during the boot process. The USB flash drive then serves as a hardware key, so to speak, which is required to boot the computer. Some card readers can also be used on Linux, but integration into the encryption software requires manual work.

The Arch Linux Wiki describes a wide range of configuration variants (see *https://wiki.archlinux.org/title/Dm-crypt/Encrypting_an_entire_system*).

18.19.4 Emergency Plan

The nightmare of all Linux users is that the computer will no longer boot and thus the data cannot be accessed. Encryption makes this scenario even more dramatic.

Of course, you have a backup, don't you?

In addition, it's a good idea to print the output of `lsblk` and keep it in a safe place. This will at least give any Linux-savvy helper some guidance about your computer's setup.

It's even better if you're able to help yourself. To do this, boot your computer with a live system, such as a USB flash drive with the current Fedora or Ubuntu installation image. In a terminal window, you can then use `cryptsetup luksOpen` to unlock the partition with the encrypted data. For this, you need your encryption password. (If you have forgotten the password, there is currently no way of recovering your data.) Then use `pvscan`, `vgscan`, and `lvscan` to determine the key data of the LVM system. If the commands do not produce any results, you may need to run `modprobe dm-mod` first. And if the `pv`, `vg`, and `pvscan` commands are not available at all, you must install the `lvm2` package.

Finally, `mount` the file system with your files into the local directory tree. This allows you to access your data and back it up to a USB hard drive or network directory.

Of course, you should try out the entire process in a quiet time before a disaster actually occurs. The following commands provide an example of the sequence of commands in the live system. Of course, if you try this out on your computer, the device, VG, and LV names will be different:

```
root# cryptsetup luksOpen /dev/nvme0n1p2 mycrypt
Enter passphrase for /dev/nvme0n1p2: *************
root# pvscan
  PV /dev/mapper/mycrypt   VG myvg lvm2 [<1.62 TiB / 76.72 GiB free]
  Total: 1 [<1.62 TiB] / in use: 1 [<1.62 TiB] / in no VG: 0 [0   ]
root# vgscan
  Found volume group "myvg" using metadata type lvm2
root# lvscan
  ACTIVE            '/dev/myvg/home' [1.46 TiB] inherit
  ACTIVE            '/dev/myvg/root' [80.00 GiB] inherit
root# mkdir /mnt/mydata
root# mount /dev/myvg/home /mnt/mydata
```

If your computer has a hardware defect and will not boot, you should remove the hard disk and install it in another computer. After that, the same procedure described previously applies. (You are out of luck with notebook computers with soldered SSDs.)

Chapter 19
Grand Unified Bootloader

The *Grand Unified Bootloader* (GRUB) is the first program to be started when the computer gets switched on. For BIOS computers, you can then choose between Linux and Windows from a menu. On EFI computers, this selection function no longer plays a major role. However, GRUB also takes over the task of starting Linux itself and passing various parameters to the kernel there.

Usually, GRUB is set up during the installation of the distribution and then simply works. The background information presented here is especially relevant if the boot process gets stuck for some reason.

This chapter focuses on the GRUB 2.0 version that has been available since 2012 and its use on EFI machines. At the end of the chapter, you will also find a short section on the alternative boot method, systemd-boot, which is currently only used in a few distributions, including Pop!_OS and, depending on the installation, Arch Linux.

19.1 Basic Principles of GRUB

Components and packages

The files required for GRUB are often distributed across several packages: On Debian and Ubuntu, `grub-common` contains platform-independent configuration files and commands. The EFI-specific files are located in the `grub-efi-amd64` package.

On Fedora, `grub2-efi` contains the EFI-compatible version of GRUB. Scripts from the `grubby` package update the GRUB configuration after kernel updates.

GRUB uses the `os-prober` package depending on the configuration. The command of the same name searches for operating systems on all accessible partitions. The result of `os-prober` is included in the automatically generated GRUB menu in some distributions.

19.1.1 EFI System Startup

Before we go into the details of GRUB installation and configuration, it's useful to get an idea of what happens during the boot process. The boot process differs greatly depending on whether your computer is running a traditional BIOS or the more modern EFI.

Parallel installations

The *Extensible Firmware Interface* (EFI) simplifies the parallel installation of multiple operating systems. While the BIOS basically only provided for the installation of *one* operating system and every parallel installation required a boot manager, EFI supports the installation of multiple operating systems by itself. Each operating system can store its own bootloader in its own directory within the EFI partition. When the computer starts, EFI automatically searches for the bootloader of the default operating system and executes it. The default operating system is usually the one that was last installed. If a special key combination is pressed while the computer is booting, EFI displays a menu of all bootloaders. There you can set the default operating system.

In combination with EFI, GRUB thus only has to boot Linux. The second function of GRUB—namely, the selection between different operating systems—is actually redundant in conjunction with EFI. However, because the GRUB menu has proven to be useful, it is still often displayed. This results in the starting routes, which are indicated by dashed lines in Figure 19.1.

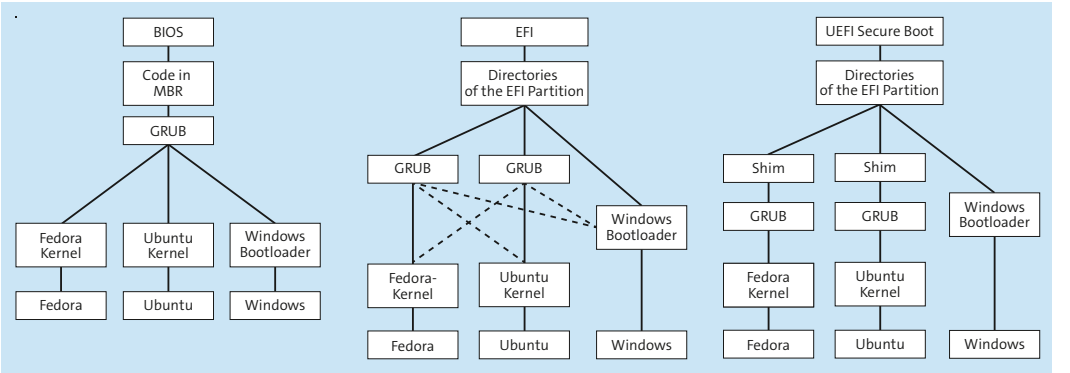


Figure 19.1 Boot Process for Three Operating Systems with BIOS, EFI, and UEFI Secure Boot

EFI and GRUB

There are different variants of GRUB for EFI and for BIOS computers. The installation programs of common distributions automatically take care of installing the correct version. However, this requires that the installation program is run in EFI mode and not in BIOS mode! (Some mainboards support both variants.)

EFI partition

The GRUB code is not installed in the MBR on EFI computers, but in a directory of the EFI partition. This is a special partition with a VFAT file system. The partition must be marked by a special UID: 0xEF (MBR) or C12A7328-F81F-11D2-BA4B-00A0C93EC93B (GPT). If you set up partitions manually using `parted`, you must mark the EFI partition with the `esp` flag (for *EFI system partition*, set with `set N esp on`). You must also make sure that the EFI

partition in the `/boot/efi` directory will be mounted in the Linux file system. The installation programs of most Linux distributions take care of these details themselves. The exceptions include Arch Linux, where manual work is required at this point.

Microsoft recommends setting up the EFI partition as the first partition on the hard disk, although the EFI standard does not require this. It used to be common practice in the past to set the EFI partition as relatively small (100 to 200 MiB). But if possible, you should provide significantly more space because with some boot systems, not only GRUB but also the Linux kernel and additional files are stored in the EFI partition. The files for performing firmware updates must also be stored in the EFI partition. Windows 10 and 11 now use at least 500 MiB, and Fedora also adheres to this minimum value starting with version 39.

Linux startup

To actually start Linux, the bootloader must load the Linux kernel file into the memory and execute it. Usually the filename of the kernel file is `/boot/vmlinuz`. The last letter *z* indicates that the kernel is compressed. Thus, the bootloader must be able to load a complete file from a Linux file system.

Kernel parameters

Several parameters are usually passed to the kernel, but there must be at least one: the device name of the system partition or the UUID of the root file system (e.g., `root=/dev/nvme0n1p2` or `root=UUID=8164c15f...`). This tells the kernel which partition is the system partition. As soon as the kernel is running, it passes control to the init system. This program is responsible for initializing Linux and is described in detail in Chapter 20. For example, it takes care of starting all network services.

Access to kernel modules

The Linux kernel is *modularized*. This means that the kernel itself contains only relatively basic functions. Additional functions for accessing certain hardware components, reading and writing various file systems, and so on are located in modules that are loaded from the file system as needed and thus extend the kernel.

For the boot process to succeed, the kernel must be able to access the system partition. If this partition is in a file system that the kernel does not directly support, or if the partition is on a SCSI disk for which the kernel does not contain a hardware driver, a chicken-and-egg problem occurs: the kernel cannot access the file system and therefore cannot load the modules it would need to read files of the file system...

Initrd file

The solution to the problem is that the boot loader loads not only the kernel, but also a *initrd* file. This is a special file that contains all the kernel modules required for the startup process. The file is temporarily available to the kernel as a RAM disk; that is, the kernel can load the required modules from the RAM disk immediately after startup (*initrd* is short here for

initial RAM disk). The name of the initrd file usually is `/boot/initrd` or `/boot/initrd.gz`. Most distributions provide tools to create an initrd file that matches the kernel.

GRUB installation and configuration

When we talk about *software installation* in this book, we're usually referring to the installation of a program package on the hard disk. However, in this chapter, different rules apply. *Installing GRUB* refers to the process of writing the GRUB startup code to the EFI partition and changing some EFI parameters. Also, GRUB configuration scripts are executed in the `/etc/grub.d/` directory. These scripts create the actual GRUB-2 configuration file, `/boot/grub2/grub.cfg`.

19.1.2 UEFI Secure Boot

The *Secure Boot* EFI extension has made the boot process even more complicated. When this EFI extension is active, the EFI starts only those programs that are signed with a key that EFI can recognize.

However, most mainboards support only one key—the one from Microsoft! Fortunately, Microsoft offers a signing service for Linux distributors and other companies. This makes it possible in principle to have a bootloader signed so that EFI is ready to boot it.

However, there are conditions associated with the signing service: participants must commit to securing the system to be booted against tampering so that Secure Boot lives up to its name and does not load malware, but only the original Linux kernel of the respective distribution.

To keep the code signed with the Microsoft key as small as possible, the distributors decided to take an intermediate step. EFI first starts the signed Shim program. The only task of this tiny program is to ensure the signature chain and start GRUB. GRUB then continues the boot process as before (see Figure 19.1, right side).

In current versions of Debian, Fedora, SUSE, and Ubuntu, GRUB, the kernel, and the kernel modules are each signed with distribution-specific keys, resulting in a closed signature chain from EFI to the last kernel module.

Note that there are still various distributions that do not support Secure Boot. In that case, a boot is only possible if the Secure Boot functions are deactivated in EFI.

The price of security

Unfortunately, the strict Secure Boot implementations of Fedora, openSUSE and Ubuntu result in some restrictions from a Linux perspective:

- The kernel modules of the proprietary NVIDIA graphics drivers cannot be loaded. Because these modules have been compiled directly by NVIDIA, it's impossible to sign them with a distribution-specific key. The

same restriction applies to all other kernel modules of proprietary hardware drivers, of course.

- It is not straightforward to compile and set up your own kernel. This kernel must also be signed, but only the distributor has the necessary key. One way out of this dilemma is to use the *Machine Owner Key* (MOK) method developed by SUSE. This method stores another key provided by yourself in Shim in addition to the SUSE key. A self-compiled kernel and its modules can then also be signed with this key. The MOK concept has now been adopted by other distributions as well.
- GRUB can only boot another Linux distribution if it is signed with the same key.
- The `kexec` and `kdump` kernel functions cannot be used.

The easiest way to get around these restrictions is to disable Secure Boot.

What Does Secure Boot Protect Against?

Secure Boot is intended to prevent malware from being loaded during the boot process, which subsequently evades all further security measures, such as virus scanners. However, such attacks have been rare in recent decades.

19.1.3 BIOS System Boot

Computers without EFI are almost extinct. However, a BIOS system boot without EFI is still common so long as Linux is running in a virtual machine. For this reason, this section very briefly describes this special case.

Virtual machines have a decisive advantage over "real" computers: there is no need to install multiple operating systems in parallel. The usual battle between Linux and Windows over the boot sector (i.e., the first sector of the hard disk) is therefore a thing of the past.

From BIOS via GRUB to the kernel

When you start an old PC or virtual machine, the BIOS is initialized first. Then the BIOS loads the contents of the first sector of the first hard disk into the memory and executes this code. This special sector of the hard disk is referred to as the *master boot record* (MBR). After a Linux installation, the MBR contains the code of GRUB.

The MBR is only 512 bytes in size—too small to store the entire bootloader program. That's why the MBR contains just enough code to load the rest of the bootloader from the disk. Accordingly, the GRUB code is divided into two or three parts: `stage1` is located in the MBR and has the task of loading

the first sectors of `stage1_5` or `stage2`. `stage1_5` contains additional code for accessing files in various file systems, and `stage2` contains the actual bootloader.

GRUB then displays a menu that offers a choice between the current and older kernel versions. After a few seconds, the first entry will be activated automatically. So GRUB loads the kernel and starts it, passing the initrd file and kernel parameters.

BIOS and GPT

Writing GRUB directly to the first sectors of the disk is now frowned upon. Especially on hard disks or SSDs using GPT, it's therefore recommended to provide a separate BIOS GRUB partition as the location for GRUB. The partition does not need to be large; 1 MiB is enough. The installation programs of most distributions take care of this automatically. For manual partitioning, you can set up a partition using `parted` and mark it with `set <n> bios_grub on`.

19.1.4 The initrd Files

Chicken or egg?

Linux uses a modularized kernel. Many additional functions—for example, for hardware components or for access to certain (possibly encrypted) file systems, RAID arrays, or LVM partitions—are not located directly in the kernel, but in modules. However, this causes a problem at system startup: How should the kernel load a module if it can't yet access the file system? For this reason, the modules required for the immediate startup are packed into an initial RAM disk. GRUB passes the corresponding initrd file to the kernel (`initrd` keyword in the GRUB configuration file).

initrd:internal details

The kernel and the initrd file are stored in the `/boot` directory.

The initrd file must contain kernel modules whose version exactly matches the version of the kernel. For this reason, every time a new kernel is installed or self-compiled, a matching initrd file must also be created. In the case of a kernel update, this is done by the update program. If, on the other hand, you install the new kernel yourself, you must also take care of the initrd file yourself.

The term *initrd files* is actually incorrect for most current distributions. These are in fact `initramfs` files, the structure of which is described ahead. However, because both the GRUB options and the commands for creating the files use the term `initrd` and the kernel interprets the file correctly despite the incorrect designation, I also stick with this designation in this book—against my better judgment, so to speak.

The initrd file is not always mandatory. If your kernel contains all the components that are required during the boot process, the boot will succeed

even without an initrd file. For this purpose, however, the kernel must be compiled appropriately—and this is not the case in most distributions. Unfortunately, the generation of initrd files is not standardized, and each distribution uses its own tools. Not only do the initrd files contain kernel modules, but also scripts for hardware initialization and possibly a minimal rescue system so that rescue work can be carried out even if the system partition cannot be mounted.

update-initramfs (Debian, Ubuntu)

On Debian and Ubuntu, the `update-initramfs` script is provided for generating and administrating the initrd files. In the simplest case, you just need to specify the `-u` option to update the initrd file of the latest installed kernel version. If you want to update the initrd file for a different kernel version, you must specify the version number with `-k`. `-k all` updates the initrd files for all installed kernel versions.

With the `-c` or `-d` options, `update-initramfs` creates a new initrd file or deletes an existing initrd file. In this case, specifying the kernel version via `-k` is mandatory:

```
root# update-initramfs -c -k 6.2.0-20-generic
update-initramfs: Generating /boot/initrd.img-6.2.0-20-generic
```

Behind the scenes, `update-initramfs` uses the `mkinitramfs` script to generate initrd files. The basic configuration is carried out in `/etc/initramfs-tools/initramfs.conf` and by the files in `/etc/initramfs-tools/conf*.d`. In addition, all modules listed in `/etc/initramfs-tools/modules` are added to the initrd file (one module per line). `mkinitramfs` generates rather large initrd files with countless kernel modules in the default configuration with `MODULES=most` in `initramfs.conf`. If you call `mkinitramfs` directly, you must at least pass the name of the new initrd file (`-o` option). If the initrd file is not to be generated for the current kernel version, you should additionally specify the relevant version:

```
root# mkinitramfs -o myinitrd 6.2.0-20-generic
```

dracut (Fedora, Red Hat, SUSE)

Fedora, RHEL, and SUSE/openSUSE use `dracut` to generate the initrd file. `dracut` is automatically executed with every kernel update. The `dracut` command takes into account the settings from `/etc/dracut.conf`. To manually create an initrd file for a self-compiled kernel 6.2.14 in the `/boot/vmlinuz-6.2.14` file, you need to run the following command:

```
root# dracut /boot/initramfs-6.2.14-300.fc38.x86_64.img
```

Dracut creates compact initrd files that contain only those kernel modules which are active during operation. This can lead to boot problems after

hardware extensions. For this reason, there is a special initrd file for the rescue system with many more modules. If the ordinary boot process fails after a hardware rebuild, you must boot the rescue system and then run the following command:

```
root# dracut --regenerate-all --force
```

mkosi Fedora is currently experimenting with a new tool for generating initrd files. `mkosi-initrd` is to be shipped in Fedora 39 for the first time in parallel with `dracut`. If the program proves successful, it could replace `dracut` in the future. The main advantage of `mkosi-initrd` is said to be that the tool is easier for distributors to maintain and configure. For more information, visit *https://fedoraproject.org/wiki/Changes/mkosi-initrd*.

Kernel updates When a distribution performs a kernel update and this changes the name of the kernel file, the GRUB menu file must also be updated accordingly and an initrd file created to match the new kernel. All common distributions do these tasks automatically as part of the update management so that the new kernel is automatically used the next time the computer is started. For emergencies (such as a defective update), the previous kernel can also be selected via a GRUB menu entry.

19.1.5 The Future of the Boot Process

If you want to look at it positively, the boot process of Linux is mature and stable. Linux thought leaders like Lennart Poettering are more critical in this regard and have pointed out various shortcomings in recent years:

- Even if UEFI Secure Boot is involved, not all aspects of the boot process are perfectly secured. In particular, the initrd file generation just described is a possible point of attack for hackers.
- Linux updates are applied to an active system. This is associated with risks, especially since often neither the running processes nor the kernel are restarted. It would be safer to apply updates at the start of the boot process or when shutting down. (However, this would cause noticeable downtime for updates, which is particularly unwelcome in the server arena.) An even better solution would be two root file systems that are alternately updated and used for booting (a so-called A/B system).
- Linux installations with an encrypted file system have two passwords, one for the file system and a second one for the login. This is not elegant.
- Security features like Trusted Platform Module (TPM) as well as biometric authentication methods (e.g., fingerprint sensors) are poorly or not at all supported in Linux, and certainly not supported as part of the boot process.

The boot process of any current Apple or Android smartphone is now better and more consistently secured than that of Linux. Modern notebook computers that run Windows or macOS are also ahead of Linux in this respect, even though these operating systems struggle with various security problems of their own.

A complete rebuild of the boot process for Linux is currently not in sight. This is not least due to the fact that Linux's share of the desktop market is small. The motivation and budget for further development of server features is considerably greater.

Regardless of this, the Linux community is definitely thinking about how the boot process can be improved. It remains to be seen whether and to what extent these ideas will actually be implemented. I recommend that technically minded Linux users interested in this topic take a look at the following pages:

- *https://0pointer.net/blog/brave-new-trusted-boot-world.html*
- *https://0pointer.net/blog/authenticated-boot-and-disk-encryption-on-linux.html*
- *https://blog.davidbyrne.io/2018/08/16/linux-ab-partitions*

19.2 Operating GRUB (User View)

After a Linux installation, a menu for selecting the preferred operating system appears when the computer is restarted (see Figure 19.2). The appearance of GRUB can vary greatly depending on the configuration. Some distributions start GRUB in graphics mode and thus take away the spartan look of the program. To be able to use various additional functions of GRUB, you must exit the graphics mode by pressing the Esc key.

Other distributions avoid displaying the GRUB menu at all if possible. For example, if Ubuntu is the only operating system installed on a computer, you will only get to see the GRUB menu if you press a key during the computer startup (⇧ is enough).

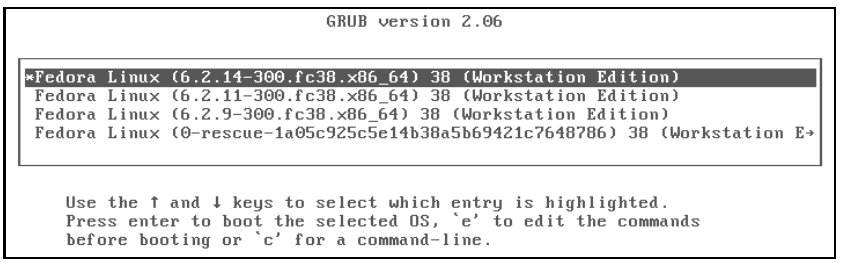


Figure 19.2 Minimalistic GRUB Menu

Password

GRUB can be secured by a password. In this case, you can use GRUB's interactive functions only after pressing P and then specifying the password.

Passing kernel boot options

Unless the GRUB menu is secured by a password, you can change the GRUB menu entry just selected with the cursor keys by pressing E (*edit*). A few lines are then displayed, which describe in a rather idiosyncratic syntax how the operating system is to be started.

GRUB's editing functions are mainly used to specify additional kernel options before Linux starts—for example, to work around hardware problems. To do this, you want to look for a line that resembles the following pattern:

```
linux /pfad/vmlinuz-n.n.n root=/dev/xxx [various options]
```

At the end of this line, you can add or change parameters. Ctrl+X or F10 then starts Linux with the changed parameters. However, the changes are not saved!

Changing the menu permanently

GRUB reads the boot menu from the `grub.cfg` file, which is stored in different locations depending on the distribution. The file contains commands that describe the GRUB menu items. So if you want to change the GRUB menu permanently, you need to start Linux and change the GRUB menu file (see the following section).

19.3 GRUB Configuration

grub.cfg

The GRUB menu is defined by the `/etc/grub/grub.cfg` or `/etc/grub2/grub.cfg` file. `grub.cfg` in the EFI directory contains a few lines of text that can be read by GRUB to indicate where the "correct" `grub.cfg` file is located. Also, on Fedora and RHEL, the `/etc/grub2-efi.cfg` link points to `/etc/grub2/grub.cfg`.

/etc/grub.d/

Manual changes to `grub.cfg` are *not* provided! The access rights of this file are therefore set to *read-only*. If you want to modify the GRUB menu, you must modify the underlying configuration files:

`/etc/grub.d/*`	(general GRUB configuration files)
`/etc/default/grub`	(additions: Debian, RHEL, SUSE, Ubuntu)
`/etc/sysconfig/grub`	(additions: Fedora)

update-grub

To regenerate the GRUB menu file, `grub.cfg`, after changes to these files or after a kernel update, you must run one of the following commands:

```
root# update-grub                              (Debian and Ubuntu)
root# grub2-mkconfig -o /boot/grub2/grub.cfg   (Fedora, RHEL, SUSE)
```

Example for grub.cfg

The resulting `grub.cfg` file looks something like the following example. The following listing of an automatically generated version of `grub.cfg` on Ubuntu is heavily shortened for space reasons. By the way, don't be confused by the `insmod ext2` line: this GRUB module is responsible for all `ext` file systems, including `ext3` and `ext4`. Further explanations of the syntax in `grub.cfg` follow later in this section:

```
# Example for /boot/grub/grub.cfg
# from /etc/grub.d/00_header
# Variable management code ...
# Definition of various functions: savedefault, recordfail, load_
video
# Find font files ...

# from /etc/grub.d/05_debian_theme
# Set colors for text mode ...

# from /etc/grub.d/10_linux
# Definition of the gfxmode function ...

# Start Ubuntu
menuentry 'Ubuntu' ... {
  recordfail
  load_video
  gfxmode $linux_gfx_mode
  insmod gzio
  insmod part_gpt
  insmod ext2
  set root='hd0,gpt2
  search ... --set=root 508de33b-b057-4e5f-b936-db8eb1f5ae51
  linux   /boot/vmlinuz-6.2.0-20-generic root=UUID=4d68... ro quiet
splash  $vt_handoff
  initrd  /boot/initrd.img-6.2.0-20-generic
}

# Menu with old Ubuntu versions and the rescue system
submenu 'Advanced options for Ubuntu' ... {
  menuentry 'Ubuntu, with Linux 5.11.0-13-generic' ... { ... }
  menuentry 'Ubuntu, with Linux 5.11.0-13-generic (recovery mode)'
... {
    ...
    linux   /boot/vmlinuz-5.11.0-13-generic root=UUID=4d68... ro
recovery  nomodeset
```

```
    ...
  }
}
```

Recreating grub.cfg

On Debian and Ubuntu, the `update-grub` command creates a new version of `grub.cfg`. On Fedora and openSUSE, you need to run `grub2-mkconfig -o /path/to/grub.cfg` instead. The Debian- or Ubuntu-specific `update-grub` script also contains only this command.

`grub2-mkconfig` analyzes the configuration files or scripts described ahead. Among other things, GRUB menu entries are created for all kernel files in `/boot`. In addition, all accessible partitions are scanned. If they contain other operating systems, GRUB menu entries will also be created for them. For this reason, `update-grub` takes a while to run on machines with many partitions. `grub2-mkconfig` is run automatically with each kernel update and ensures that the latest Linux kernel is included in the GRUB menu.

Basic GRUB settings

The `/etc/default/grub` file contains some global GRUB settings. Remember that changes made here will not take effect until you regenerate `grub.cfg`! On Ubuntu, the configuration file contains the following settings:

```
# file /etc/default/grub (Ubuntu)
GRUB_DEFAULT=0
GRUB_TIMEOUT=10
GRUB_DISTRIBUTOR=`lsb_release -i -s 2> /dev/null || echo Debian`
GRUB_CMDLINE_LINUX_DEFAULT="quiet splash"
GRUB_CMDLINE_LINUX=
```

The default Fedora configuration looks as follows:

```
# file /etc/sysconfig/grub (Fedora)
GRUB_TIMEOUT=5
GRUB_DISTRIBUTOR="$(sed 's, release .*$,,g' /etc/system-release)"
GRUB_DEFAULT="saved"
GRUB_DISABLE_SUBMENU="true"
GRUB_TERMINAL_OUTPUT="console"
GRUB_CMDLINE_LINUX="rhgb quiet"
GRUB_DISABLE_RECOVERY="true"
GRUB_ENABLE_BLSCFG="true"
```

In the following sections, I explain the meaning of some parameters, even those that are not active by default:

- `GRUB_DEFAULT` specifies which GRUB menu item should be selected by default. The `saved` setting means that the last selected menu item will be activated. However, this only works if the GRUB files are located in an

ordinary partition! If instead LVM or RAID is involved, GRUB cannot save any environment variables after the menu selection.

Another option is to assign the `menuentry` string of the desired menu item to `GRUB_DEFAULT`. However, you must be careful to keep the spelling exact.

- `GRUB_TIMEOUT=n` specifies how many seconds GRUB waits for a menu item to be selected. If this time elapses without user input, then GRUB starts the selected operating system. The time set here only comes into effect when the GRUB menu appears at all.
- The `GRUB_DISTRIBUTOR` variable is analyzed by the `10_linux` script, which I describe ahead. It specifies the name of the current distribution, such as `Ubuntu` or `Fedora`.
- `GRUB_CMDLINE_LINUX` and `GRUB_CMDLINE_LINUX_DEFAULT` are also considered by `10_linux` and specify which options must be passed to the kernel. The `GRUB_CMDLINE_LINUX` options apply to every boot; the `GRUB_CMDLINE_LINUX_DEFAULT` options are added additionally for standard boot, but not for recovery mode.
- By default, the GRUB menu is displayed in graphics mode at a resolution of 640 × 480 pixels. If you want a higher resolution, you can set it using `GRUB_GFXMODE`. If you want to dispense with the graphics mode altogether, you can activate the `GRUB_TERMINAL="console"` (Debian/Ubuntu) or `GRUB_TERMINAL_OUTPUT="console"` (Fedora/RHEL) setting provided for this purpose. Both variables are analyzed by the `00_header` script. In the default setting, there is no clearly visible difference between text and graphics mode, but Unicode characters can only be displayed in graphics mode.
- GRUB normally passes the root directory as a UUID number to the Linux kernel to be started. If you prefer to specify the device name instead (e.g., `/dev/sda1`), you need to enable the `GRUB_DISABLE_LINUX_UUID="true"` line. This setting applies only to the start of the active distribution (`10_linux` script), not to other distributions.
- `update-grub` or `grub2-mkconfig` usually also creates menu entries for starting Linux in recovery mode. This starts Linux in single-user mode and without displaying a splash screen. With `GRUB_DISABLE_RECOVERY="true"`, no recovery entries are created.
- `GRUB_ENABLE_BLSCFG="true"` causes GRUB to analyze the configuration files in `/boot/loader/entries` when creating the boot menu. Currently, this concept is only implemented in Fedora and RHEL. It causes `boot.cfg` to be much shorter because the actual boot entries are missing there. Their contents are loaded from `/boot/loader/entries` only when GRUB is executed. The details are documented at *https://fedoraproject.org/wiki/Changes/BootLoaderSpecByDefault*.

What is usually not included in /etc/default/grub is the setting of the GRUB_RECORDFAIL_TIMEOUT variable. This variable controls how long GRUB displays the menu if the computer froze during the last boot process (or if GRUB thinks it did, which is not always the same thing). By default, the time span is half a minute. Especially on a server where you cannot watch the boot process, this can create the impression that the computer is no longer responding at all. A solution to that problem would be to set a shorter timeout! For example:

```
# /etc/default/grub or /etc/sysconfig/grub
GRUB_RECORDFAIL_TIMEOUT=5
```

grubby On Fedora, RHEL, and its clones, you can use the grubby command to make basic changes to the GRUB configuration *without* modifying the GRUB configuration files in an editor. The following command adds the selinux=0 kernel option to all GRUB menu entries, thus disabling SELinux from the next boot:

```
root# grubby --update-kernel ALL --args selinux=0
```

The second command undoes this change and removes selinux (with whatever setting) from the list of kernel parameters:

```
root# grubby --update-kernel ALL --remove-args selinux
```

19.3.1 Automatic Generation of grub.cfg

The basic idea of the GRUB configuration is that you yourself only specify the basic data of the GRUB configuration. The final configuration file, grub.cfg, is generated automatically by a script, taking these specifications into account. The key point here is that the GRUB scripts take into account all installed kernel versions (important after a kernel update). Many distributions also include boot entries for other operating systems (like os-prober; discussed ahead).

The /etc/grub.d directory contains several executable script files for this purpose. When a new version of grub.cfg is to be created, update-grub executes all scripts contained in grub.d in order. The result, the standard output of the scripts, ends up in grub.cfg. The script-based approach unfortunately makes the configuration files quite confusing. Statements like cat << EOF are repeatedly embedded in the actual code, which direct all subsequent lines up to the abbreviation EOF to the standard output. These lines themselves often contain script code that is only analyzed by GRUB at system startup.

Linux boot entries

The two most interesting scripts are `10_linux` and `30_os-prober`. `10_linux` provides two GRUB menu entries for each kernel version in the `/boot/` directory: one for an ordinary boot and a second one for a recovery boot in single-user mode and without a splash screen. The menu items are sorted by version numbers, with the latest version at the top of the list.

os-prober

`30_os-prober` calls the `os-prober` script. It provides a list of all operating systems on all accessible hard disk partitions. Menu entries are then created for each of these operating systems, whereby Linux distributions draw on the possibly existing GRUB configuration. This causes countless scripts to be called, all of which are part of the `os-prober` package. `os-prober` is actually a relic from the BIOS past, when GRUB was still responsible for starting all operating systems on a computer. Meanwhile, the EFI can take over this task. Some distributors have therefore decided to remove `os-prober` (such as Ubuntu from version 22.04 and Debian from version 12). This is why `grub.cfg` only contains entries to start Ubuntu or Debian, but not those to start Windows or other Linux distributions installed on the computer. The main advantage of this change is that recreating `grub.cfg` is now much faster.

As I said, normally you don't need GRUB menu entries for other operating systems anyway because you can use EFI to select the operating system. But in the event that something goes wrong with the EFI configuration, the additional GRUB entries are a good plan B, which I have often resorted to in the past in the event of boot problems.

Currently, the entire `os-prober` infrastructure is still included. It's therefore quite easy to reactivate `os-prober`. To do this, you need to add the following line at the end of `/etc/default/grub`:

```
# in /etc/default/grub
GRUB_DISABLE_OS_PROBER=false
```

Then recreate `grub.cfg` by calling `update-grub`.

19.3.2 Syntax and Internal Details

A complete syntax description of all keywords allowed in `grub.cfg` is impossible to provide here due to space limitations. I therefore limit myself in the following sections to the most important keywords that appear in the examples presented here. The official GRUB manual (the PDF version comprises 150 pages) can be found at the following address: *https://www.gnu.org/software/grub/manual*.

Variables

You can use `set varname=value` to perform variable assignments. To read variables, you should use the `$varname` notation. When you run GRUB

commands interactively, `echo $varname` displays the contents of a variable, and `set` returns all defined variables.

Partitions

GRUB has its own terminology for naming hard disks and the partitions they contain. For example, `hd0,1` denotes the first MBR partition of the first disk or SSD, while `hd1,gpt8` denotes the eighth GPT partition of the second disk.

Handling UUIDs (search)

As I explained in Chapter 18, the use of numbered devices for disks is prone to errors. That is why the use of UUIDs has also become established in GRUB:

```
search --no-floppy --fs-uuid --set root 724f...
```

The preceding statement searches for a file system with the specified UUID, such as `724f`. If the search is successful, GRUB saves the corresponding partition name in the `root` variable because of the `--set` option.

Modules

With `insmod name`, GRUB loads extension modules with additional functions at runtime. GRUB looks for the `name.mod` module files in the `/boot/grub[2]` directory in the partition set by the `root` variable. Important modules include `part_msdos` and `part_gpt` (read partition tables), `ext2` (file systems `ext2` through `ext4`), `raid`, `raid5rec`, `raid6rec`, and `mdraid` (software RAID), `lvm`, `gfxterm` (graphical console), `vbe` (graphics system), and `jpeg`, `tga`, and `png` for reading graphics files.

19.3.3 GRUB Menu Items

menuentry and submenu

GRUB menu entries are introduced with the `menuentry` keyword. The following text is in quotation marks and may contain international characters. GRUB also supports submenus. When operating GRUB, pressing Esc from a submenu will return you to the main menu if necessary:

```
menuentry "Linux" {
  start commands ...
}
submenu 'Submenu' {
  menuentry 'Entry 1' { ... }
  menuentry 'Entry 2' { ... }
}
```

Launch Linux

The minimal version of a menu entry for starting Linux looks like the following example:

```
menuentry "Linux" {
  set root=(hd0,3)
  search --no-floppy --fs-uuid --set root 724f...
```

```
  linux   /boot/vmlinuz-n.n.n    root=... ro quiet splash
  initrd  /boot/initrd.img-n.n.n
}
```

`set root` specifies the partition where the kernel and initrd files are located. Alternatively, `search` looks for the partition whose file system has the specified UUID. The `linux` and `initrd` keywords specify filenames relative to the partition.

The parameters specified with `linux` are passed to the kernel. You absolutely need `root` to specify the system partition, and `ro` so that access to the system partition is initially read-only. All other parameters are dependent on the distribution.

Separate boot partition

Many Linux distributions (such as Fedora and RHEL) provide for a separate boot partition during the installation. In this case, you can use `set root` to specify the boot partition. In turn, this means that `linux` and `initrd` do not need to specify the boot directory:

```
menuentry "Linux - With separate boot partition" {
  set root=(hd0,2)
  search --no-floppy --fs-uuid --set root 59ac...
  linux   /vmlinuz-n.n.n     root=... ro quiet splash
  initrd  /initrd.img-n.n.n
}
```

Boot directory in RAID partitions or logical volumes

If the partition with the kernel and initrd files is part of an LVM system and/or a software RAID, you must load the corresponding GRUB modules. For RAID-5 or RAID-6, the `raid5rec` or `raid6rec` module will be added. In `set root`, you can then specify the system partition in the notation `(lvname)` or `(mdN)`:

```
menuentry "Linux - With Software RAID" {
  insmod raid mdraid
  set root=(md0)
  search --no-floppy --fs-uuid --set root 735c...
  linux   /boot/vmlinuz-n.n.n    root=... ro quiet splash
  initrd  /boot/initrd.img-n.n.n
}
menuentry "Linux - With LVM" {
  insmod lvm
  set root=(vg1-root)
  search --no-floppy --fs-uuid --set root 6bb3...
  linux   /boot/vmlinuz-n.n.n    root=... ro quiet splash
  initrd  /boot/initrd.img-n.n.n
}
```

```
menuentry "Linux - LVM on RAID-5" {
  insmod raid raid5rec mdraid lvm
  set root=(vg1-root)
  search --no-floppy --fs-uuid --set root 8ff4...
  linux   /boot/vmlinuz-n.n.n    root=/dev/mapper/... ro quiet splash
  initrd  /boot/initrd.img-n.n.n
}
```

19.4 Manual GRUB Installation and First Aid

Usually, GRUB is set up correctly during the installation of your Linux distribution. As a result, you may make changes to the GRUB configuration or `grub.cfg`, but you do not need to touch the GRUB installation as such.

The following section is only relevant if you want to perform a manual installation of GRUB for some reason or if you need to repair a defective GRUB installation. To do this, start your computer from a USB flash drive that contains a live system. If you want to repair a Fedora installation, it's best to use a Fedora live system, and an Ubuntu live system accordingly for Ubuntu.

19.4.1 Manual Installation and First Aid for EFI PCs

Manual installation using grub-install

To manually install GRUB on an EFI machine, you can run `grub-install` or `grub2-install`. On Fedora and RHEL, you must install the `grub2-efi-modules` package beforehand:

```
root# grub-install      (Debian, Ubuntu)
root# grub2-install     (Fedora, SUSE, RHEL)
```

This command creates the `/boot/efi/EFI/distribution-name` directory. A new boot file with the `.efi` extension that contains the GRUB code is written to this directory. In addition, the `.efi` file is added to the list of EFI boot entries and placed there in the first place. `grub-install` uses the `efibootmgr` command for this purpose, which is described in more detail in Section 19.4.2.

For `grub-install` to run successfully, some prerequisites must be met:

- The GRUB configuration in the `/boot/grub[2]/grub.cfg` file must be prepared.
- The EFI partition must be mounted in the directory tree via the `/boot/efi` path.

- The `efibootmgr` command from the package of the same name must be installed.
- The `efivarfs` kernel module must be loaded or compiled into the kernel. However, `modprobe efivarfs` only succeeds if the distribution was started in EFI mode, not in BIOS mode.

Repairing GRUB in a live system (EFI)

The following listing summarizes the commands required for an EFI installation from a live system. For this purpose, you need to create a working directory (here, `/syspart`) and mount your system partition and the active `/dev`, `/proc`, and `/sys` directories there. Then use `chroot` to temporarily make `/syspart` the new root directory.

If necessary, you must include the boot file system there now. Now update the GRUB configuration and write it to the EFI partition by using `grub-install`. Don't forget to replace `/dev/sda<n>` with your own device names in the following commands:

```
root# mkdir /syspart
root# mount /dev/sda2 /syspart                    (system partition)
root# mount -o bind /dev  /syspart/dev
root# mount -o bind /proc /syspart/proc
root# mount -o bind /sys  /syspart/sys
root# chroot /syspart
root# mount /dev/sda1 /boot/efi                   (EFI partition)

root# update-grub                                 (Debian and Ubuntu)
root# grub2-mkconfig -o /etc/grub2-efi.cfg        (Fedora)
root# grub2-mkconfig -o /boot/grub2/grub.cfg      (openSUSE)

root# grub[2]-install
root# exit
```

When LVM is involved, things get even more complicated. In that case, you should do without the `bind-mount` commands, use the `arch-chroot` command instead of `chroot` (available on Fedora in the `arch-install-scripts` package), and also include `/run/lvm` in the `chroot` environment. The following commands show the basic pattern:

```
root# mkdir /syspart
root# mount /dev/mapper/<name> /syspart
root# mkdir /syspart/hostlvm
root# mount -o bind /run/lvm /syspart/hostlvm
root# arch-chroot /syspart
root# ln -s /hostlvm /run/lvm
```

You can find more tips on this in the Arch Linux forum at *https://bbs.archlinux.org/viewtopic.php?id=242594*.

If you have network access in the `chroot` environment, you can also try out the following command on Fedora or RHEL:

```
root# dnf reinstall grub2-efi shim
```

19.4.2 Changing EFI Boot Entries and Settings (efibootmgr)

How does EFI actually know which operating systems are installed or which boot entries it should display? On EFI mainboards, this information is stored in nonvolatile memory (NVRAM). Each time a new operating system is installed, a corresponding entry is stored in NVRAM once the `*.efi` file has been set up in the EFI partition.

efibootmgr

On Linux, you can read or modify the EFI data stored in this way using the `efibootmgr` command. The command assumes that the `efivarfs` kernel module is loaded or compiled into the kernel. If that's not the case, you should run `modprobe efivarfs`. This only succeeds if Linux was booted in EFI mode. If necessary, you can use a Linux live system that can be started in EFI mode for repair work!

When `efibootmgr` is run without any other options, it lists the EFI boot entries and some other parameters of the EFI bootloader. The following listing (in shortened form) was created on my work notebook, on which three Linux distributions were installed at the time:

```
root# efibootmgr

BootCurrent: 001A
Timeout: 1 seconds
BootOrder: 001A,0000,0001,001B
Boot0000* ubuntu  HD(1,GPT,717a...,0x800,0x400001)/File(\EFI\ubuntu\
shimx64.efi)
Boot0001* Fedora  HD(1,GPT,717a...,0x800,0x400001)/File(\EFI\fedora\
shimx64.efi)
Boot0010  Setup                    FvFile(721c...)
Boot0011  Boot Menu                FvFile(126a...)
Boot0012  Diagnostic Splash Screen FvFile(a7d8...)
Boot0013  Lenovo Diagnostics       FvFile(3f7e...)
Boot0014  Regulatory Information   FvFile(478c...)
Boot0015  Startup Interrupt Menu   FvFile(f46e...)
Boot0016  Rescue and Recovery      FvFile(665d...)
Boot001A* NVMe0 VenMsg(bc78..., 001c...a400)
Boot001B* NVMe1 VenMsg(bc78..., 001c...a401)
```

BootCurrent tells you that the computer was last started with boot entry 001A. The bootloader was run on the NVMe0 disk (for Arch Linux with systemd-boot; see Section 19.5). The BootOrder variable means that the four specified menu items are taken into account during a restart. The first available system is actually started after a short waiting period. (So if the NVMe0 disk has been removed in the meantime, Ubuntu will come into play on the other SSD.) During the short wait time before the first bootloader starts, a mainboard-specific key combination (on my test computer, F12) can display the EFI menu. In it all entries listed in the previous listing are displayed, but the four mentioned in BootOrder are displayed first.

If you want Fedora to start once on the next reboot, you need to specify its boot entry with -n or --bootnext:

```
root# efibootmgr --bootnext 0001
```

If you want to change the boot order permanently, you should specify the entry or a list of entries with the -o or --bootorder option:

```
root# efibootmgr --bootorder 0001,001A,0000,001B
```

The following command creates a new EFI boot entry. You can use the --disk and --part options to specify the location of the EFI partition. --loader points to the EFI code to run, usually shimx64.efi. The path specification is relative to the EFI partition (i.e., /boot/efi), and \ must be used as the directory separator. The doubling of the \ characters is necessary because the shell interprets a single \ character as a special character identifier. --label determines the name to be displayed in the EFI menu:

```
root# efibootmgr  --create --disk /dev/nvme0n1 --part 2 \
       --loader \\EFI\\ubuntu\\shimx64.efi --label ubuntu
```

Of course, you can also remove boot entries. To do this, use --bootnum to specify the number of the entry:

```
root# efibootmgr --bootnum 0001 --delete-bootnum
```

19.5 systemd-boot

In short, GRUB works as follows on modern PCs: EFI loads the GRUB files from the EFI system partition (ESP) and starts them. Then GRUB searches the Linux file system for the kernel and the initrd file and starts the kernel. And finally, the kernel takes care of the actual system startup.

The systemd-boot project, which is part of systemd (see Chapter 20), saves one step in this process: the kernel and initrd file are placed right in the ESP and started by a tiny bootloader.

The systemd-boot project evolved from the EFI boot project called *gummiboot* (German for "rubber dinghy").

This section only provides a brief overview of how systemd-boot works and the available configuration options. You can find more technical information on the following web pages:

- *https://support.system76.com/articles/bootloader*
- *https://wiki.archlinux.org/title/systemd-boot*
- *https://www.rodsbooks.com/efi-bootloaders/gummiboot.html*
- *https://www.freedesktop.org/software/systemd/man/systemd-boot.html*

Benefits and drawbacks

The obvious advantage of systemd-boot is the strong simplification and associated (rather minimal) acceleration of the boot process. Because all necessary files are located in the EFI partition, the bootloader doesn't have to deal with various file systems and all their special cases (LVM, RAID, and encryption).

Unfortunately, there are also disadvantages, and they are why systemd-boot has not gained more acceptance so far:

- EFI is mandatory; systemd-boot is not compatible with old PCs or virtual machines without EFI.
- The kernel and especially the associated initrd file take up a lot of space, typically around 50 to 100 MiB per release. For security reasons, the boot system is redundant. If there are boot problems after a kernel update, the previous kernel version can be selected in GRUB or systemd-boot. Sometimes the distribution still provides for a rescue system. This means that three kernels and three initrd files are required and the space requirement is already about 250 MiB. With parallel installation of further distributions, the space requirement multiplies.

  On notebook computers with preinstalled Windows, the EFI partition is often only 250 MiB. So even with the installation of only one Linux distribution, it can happen that the EFI partition is too small.
- sytemd-boot can only boot Linux—no other operating systems.
- By default, systemd-boot does not support UEFI Secure Boot (or only does so if you compile and sign the kernel yourself and store the key used in the EFI).

Installing Pop!_OS and Arch Linux

Currently, Pop!_OS uses systemd-boot by default on computers with EFI. For Arch Linux, you need to set up the boot system yourself during the

installation and have a choice between systemd-boot and GRUB. In both cases, it's recommended that the ESP be generously sized at 1 GiB. If the EFI partition is predefined by Windows and cannot be enlarged, it usually helps to simply set up a second EFI partition.

19.5.1 Operation

Visually, systemd-boot looks like GRUB (see Figure 19.3).

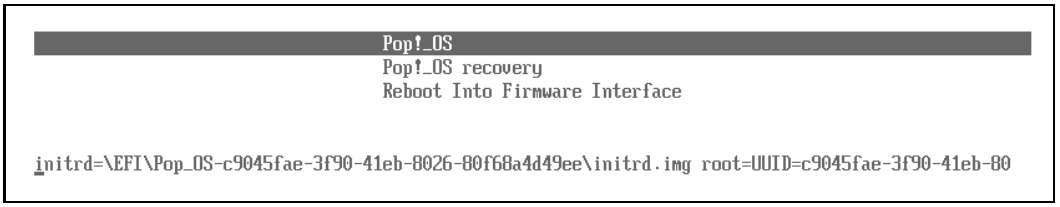


Figure 19.3 systemd-boot Options to Change Kernel Parameters

You can use the cursor keys to select the desired kernel, Enter starts Linux. E takes you to a basic editor in which you can change the options passed to the kernel. Further interventions in the boot process are not possible.

19.5.2 Configuration

During the installation of Linux, all boot files are copied directly to a subdirectory of /boot/efi. The following lines show the files of Pop!_OS:

```
root# cd /boot/efi
root# tree
EFI
   BOOT
       BOOTX64.EFI
   Pop_OS-1f37...
       cmdline
       initrd.img
       initrd.img-previous
       vmlinuz.efi
       vmlinuz-previous.efi
   Recovery-7be5...
       initrd.gz
       vmlinuz.efi
   systemd
      systemd-bootx64.efi
loader
```

```
entries
   Pop_OS-current.conf
   Pop_OS-oldkern.conf
   Recovery-7be5....conf
loader.conf
```

loader.conf The `*.conf` files in `/boot/efi/loader/entries` determine the systemd boot menu entries. The only file to which changes are planned is `loader.conf`. This file specifies how long you have to select the kernel and which kernel is to be started by default. By default, Pop!_OS lacks the `timeout` line; the boot menu is then not displayed at all:

```
# file /boot/efi/loader/loader.conf
default Pop_OS-current
timeout 5
```

bootctl The `bootctl` command provides a summary of the current configuration without any further options. To reinstall or repair systemd-boot, you must run the command as follows:

```
root# bootctl --path=/boot/efi install
```

This copies `systemd-bootx64.efi` and adds it to the list of EFI boot entries. You need to take care of the kernel and initrd files as well as the configuration files yourself.

On Pop!_OS, `update-initramfs` is configured so that each time a new initrd file is created as part of a kernel update, the relevant EFI files are also updated. If necessary, you can trigger this process manually:

```
root# apt install --reinstall linux-generic linux-headers-generic
root# update-initramfs -c -k all
```

Chapter 20
The Init System

This chapter describes the operations that take place from kernel startup to login. The kernel starts the init system as the first process. The init system then takes care of the basic configuration of the system, the mounting of file systems, and the starting of countless network services and daemons. The same init system is also responsible for shutting down the computer properly.

In recent years, systemd has established itself as *the* init system. That was not always the case. Until ten years ago, the Init-V system, which originated in Unix times, was commonly used, and for some time it looked as if the Upstart system developed by Ubuntu would have a chance to prevail.

For this reason, this chapter focuses largely on systemd. Init-V is only used marginally because there are still some services whose startup scripts have not been converted to systemd. While a compatibility layer of systemd takes care of such cases, it still doesn't hurt if you have at least a rough idea of Init-V concepts.

The chapter also briefly describes the `shutdown`, `reboot`, and `halt` commands and goes into some distribution-specific details in the init processes of Fedora/RHEL, Debian/Ubuntu, Raspberry Pi OS, and SUSE/openSUSE.

20.1 systemd

systemd is a comparatively new init system developed by Red Hat employee Lennart Poettering and was first used in Fedora in 2011. Most current Linux distributions have followed its example.

Basic principles

As usual on Linux, systemd is configured using text files. systemd itself, however, is a compiled program. This is remarkable in that traditional init systems consisted of countless shell scripts. systemd uses control groups (cgroups) to execute and monitor processes. Cgroups is a kernel function that limits the resources of a process (CPU, memory, I/O). systemd starts each process in its own cgroup. When the number of processes in this group drops to 0, systemd knows that the process has terminated or crashed and can restart it if necessary.

In an increasing number of distributions, systemd is not only responsible for the administration of system services, but also takes care of the start of

user-specific processes after a login—for example, when initializing a GNOME session.

> **Directory Locations**
>
> This section describes systemd as it is currently implemented in most distributions. However, individual distributions use different directory locations; for example, openSUSE uses `/usr/lib/systemd` instead of `/lib/systemd`.

Procedure

The following points provide a brief overview of the system startup:

- GRUB loads and starts the kernel.
- The kernel starts the systemd process. The program file is located in the `/usr/lib/systemd` directory or in `/usr/bin` in most distributions.
- Any options passed to the kernel (referred to as *boot options*) that the kernel does not know about are passed to systemd.
- systemd analyzes the configuration files in `[usr]/lib/systemd` and `/etc/systemd`, performs the initialization tasks mentioned there, and starts network services and the display manager, including the graphics system.
- The user will see the login prompt after a few seconds. In many distributions, another systemd instance is started at the user level after login to run various processes required for the desktop system.

systemd is therefore the first running process. All other processes are started either directly by systemd or indirectly by its subprocesses. If you run `pstree` in a terminal, you will immediately see the dominant role of systemd! (In some distributions, the first process seems to be `/sbin/init`, not `systemd`, but in fact `/sbin/init` is then a link to `systemd`.)

When the computer is shut down, systemd is the last process still running and takes care of terminating all other processes correctly.

System initialization

During system initialization, things are done that only need to be done once during computer startup:

- Initialize various system variables (including host and domain name)
- Enable the `/proc` file system and various other temporary file systems
- Set date and time
- Set keyboard layout for the text console
- Start the `udev` system
- Possibly enable RAID and LVM
- Check file systems

- Remount root partition in read-write mode
- Check and mount the file system of the other partitions
- Initialize network functions

20.1.1 Administration

Starting and stopping services

The central command to administrate systemd is `systemctl`. This allows you to manually start and stop services described by a `*.service` file:

```
root# systemctl start   sshd  (start SSH daemon)
root# systemctl stop    sshd  (stop SSH daemon)
root# systemctl restart sshd  (restart SSH daemon)
root# systemctl reload  sshd  (reload configuration of SSH daemon)
root# systemctl status  sshd  (determine status of SSH daemon)
```

ssh versus sshd; httpd versus apache

Almost all distributions have now adopted systemd as their init system. However, this does not mean that all configuration files are standardized!

On Fedora, RHEL, and SUSE, the service file for the SSH server is called `sshd.service`, but it is `ssh.service` on Debian and Ubuntu! Accordingly, for Debian and Ubuntu, you would need to replace `sshd` with `ssh` in all of the previously listed commands. Fortunately, an alias statement is included in `ssh.service`, which is why the examples provided here work across different distributions.

However, this is not true for every service. Sometimes you need to research the correct service name. For example, the Apache web server can be started on Fedora and RHEL using `systemctl start httpd`, but on Debian and Ubuntu via `systemctl start apache2`!

Permanently starting/stopping

`systemctl` can also be used to permanently enable/disable a service (like the former `chkconfig xxx on/off`):

```
root# systemctl enable  sshd
  Created symlink /etc/systemd/system/multi-user.target.wants/
sshd.service
               -> /usr/lib/systemd/system/sshd.service.
root# systemctl disable sshd
  Removed /etc/systemd/system/multi-user.target.wants/sshd.service.
```

So when services are activated, a new link is created, and when they are deactivated, this link is removed.

Start and Enable at the Same Time

It's often necessary to start a service and enable it permanently. Instead of `systemctl start` and `systemctl enable`, you can simply run `systemctl enable --now` with current systemd versions in such cases.

Similarly, `systemctl disable --now` corresponds to the single `systemctl stop` and `systemctl disable` commands.

List of all systemd services

When `systemctl` is called without any other parameters, it returns a list of all processes managed by systemd:

```
user$ systemctl
UNIT                    LOAD   ACTIVE SUB       JOB DESCRIPTION
sys-devices-xxx         loaded active plugged   /sys/devices/virtual/
                                                 block/dm-0
sys-devices-yyy         loaded active plugged   /sys/devices/virtual/
                                                 block/dm-1
...
-.mount                 loaded active mounted   /
boot.mount              loaded active mounted   /boot
dev-hugepages.mount     loaded active mounted   Huge Pages File System
...
chronyd.service         loaded active running   NTP client/server
colord.service          loaded active running   Manage Color Profiles
crond.service           loaded active running   Command Scheduler
cups.service            loaded active running   CUPS Scheduler
...
146 loaded units listed. Pass --all to see loaded but inactive units,
too.To show all installed unit files use 'systemctl list-unit-files'.
```

20.1.2 Targets

Targets define operating states that a Linux system should achieve. You can determine the currently valid default target by using `systemctl get-default`:

```
root# systemctl get-default
multi-user.target
```

Common default targets include `multi-user.target` for servers (full functionality, but without graphics system) and `graphical.target` for desktop operation. Targets can depend on each other; several targets can be active at the same time. You can get a list of all targets by using the following command. The result shows the active targets of a Fedora system with an active graphical user interface:

```
user$ systemctl list-units --type=target
UNIT                   LOAD   ACTIVE SUB    DESCRIPTION
basic.target           loaded active active Basic System
cryptsetup.target      loaded active active Local Encrypted Volumes
getty.target           loaded active active Login Prompts
graphical.target       loaded active active Graphical Interface
local-fs-pre.target    loaded active active Local File Systems (Pre)
local-fs.target        loaded active active Local File Systems
multi-user.target      loaded active active Multi-User System
network-online.target  loaded active active Network is Online
network-pre.target     loaded active active Network (Pre)
network.target         loaded active active Network
nfs-client.target      loaded active active NFS client services
nss-lookup.target      loaded active active Host and Network Name Lookups
nss-user-lookup.target loaded active active User and Group Name Lookups
paths.target           loaded active active Paths
remote-fs-pre.target   loaded active active Remote File Systems (Pre)
remote-fs.target       loaded active active Remote File Systems
rpc_pipefs.target      loaded active active rpc_pipefs.target
slices.target          loaded active active Slices
sockets.target         loaded active active Sockets
sound.target           loaded active active Sound Card
sshd-keygen.target     loaded active active sshd-keygen.target
swap.target            loaded active active Swap
sysinit.target         loaded active active System Initialization
timers.target          loaded active active Timers

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.

24 loaded units listed. Pass --all to see loaded but inactive units, too.
```

The `isolate` command allows you to change the current operating state. Thus, you can perform a reboot via the following command (of course, it's easier to just run `reboot`):

```
root# systemctl isolate reboot.target    (reboot computer)
```

Setting the default target

The default target is defined by the `/etc/system/system/default.target` link. Changing the default target and thus resetting this link can most easily be done using `systemctl set-default`:

```
root# systemctl set-default multi-user.target
Removed symlink /etc/systemd/system/default.target.

Created symlink from /etc/systemd/system/default.target
                  to /usr/lib/systemd/system/multi-user.target
```

Emergency target

Unlike the Init-V system, systemd does not know a single-user mode. Its role is taken over by the emergency target. This target can be enabled by adding the `single` keyword to the `linux` line of the GRUB boot entry at boot time. The system initialization is then kept to a minimum; in particular, there is no network access. Login is only possible in the first text console with the root password.

The emergency target is intended for repair work. When that is successfully completed, you can use `systemctl isolate multi-user` or `graphical` to activate the normal operating state.

Ctrl+Alt+Del behavior

If you press Ctrl+Alt+Del in a text console, the computer will restart. It is the special target file, `/lib/system/system/ctrl-alt-del.target`, that is responsible for this behavior.

20.1.3 Configuration

The configuration files for systemd are located in the `/etc/systemd/` and `/lib/systemd/` (Debian, Fedora, RHEL, Ubuntu) or `/usr/lib/system` (openSUSE) directories. In total, this involves almost 1,000 files, which means that the configuration is quite complex and at least as confusing as with the Init-V system.

Units play a central role in the systemd concept. They describe objects that are to be controlled by the init system. Not only does this include services that need to be started or stopped, but also network interfaces, `mount` directories, swap partitions, and so on.

Target files

You can use `*.target` files to connect multiple units into a group. Targets are comparable to Init-V runlevels (see Section 20.4.1), but there are usually many more targets that can relate to each other. The following lines show the `graphical.target` file:

```
# file /lib/systemd/system/graphical.target (Fedora)
[Unit]
Description=Graphical Interface
Documentation=man:systemd.special(7)
Requires=multi-user.target
Wants=display-manager.service
```

```
Conflicts=rescue.service rescue.target
After=multi-user.target rescue.service rescue.target \
  display-manager.service
AllowIsolate=yes
```

The `graphical.target` thus requires `multi-user.target` and also requires `display-manager.service`. Each target can be linked to an additional `<name>.target.wants` directory, in which further units are listed by means of files or by links to files; those units are supposed to be activated as well. For `graphical.target`, this listing contains only one entry:

```
root# cd /lib/systemd/system/graphical.target.wants
root# ls -l
 systemd-update-utmp-runlevel.service -> ../systemd-update-utmp-
runlevel.service
```

What is much more interesting is the `/lib/system/system/sysinit.target.wants` directory with various links to `*.service` files for system initialization:

```
root# ls /lib/systemd/system/sysinit.target.wants/
cryptsetup.target
dev-hugepages.mount
dev-mqueue.mount
dracut-shutdown.service
kmod-static-nodes.service
ldconfig.service
...
systemd-udevd.service
systemd-udev-trigger.service
systemd-update-done.service
systemd-update-utmp.service
systemd-vconsole-setup.service
```

Service files

Service (`*.service`) files describe which requirements must be met for starting a service and which command should be started. These files are located in the `/etc/systemd/system` and `/lib/systemd/system` directories. As an example, the following listing shows the `httpd.service` file, which is responsible for starting the Apache web server on Fedora:

```
# file /lib/systemd/system/httpd.service (Fedora)
[Unit]
Description=The Apache HTTP Server
Wants=httpd-init.service
```

```
After=network.target remote-fs.target nss-lookup.target \
  httpd-init.service
Documentation=man:httpd.service(8)

[Service]
Type=notify
Environment=LANG=C
ExecStart=/usr/sbin/httpd $OPTIONS -DFOREGROUND
ExecReload=/usr/sbin/httpd $OPTIONS -k graceful
# Send SIGWINCH for graceful stop
KillSignal=SIGWINCH
KillMode=mixed
PrivateTmp=true
OOMPolicy=continue
[Install]
WantedBy=multi-user.target
```

20.1.4 systemd at User Level

On an increasing number of distributions, systemd is also active at the user level, where it takes care of the execution of some processes after a login to GNOME. For this purpose, the `pam_systemd` PAM module starts a separate systemd process that only has the rights of an ordinary user:

```
root# ps axu | grep 'systemd '
USER      PID  ...  COMMAND
root        1       /usr/lib/systemd/systemd --switched-root --system
gdm       937       /usr/lib/systemd/systemd --user
kofler   1616       /usr/lib/systemd/systemd --user
```

By default, the `systemctl` command always refers to the system level, even if the command is executed without `root` privileges. To determine the processes started by systemd at the user level, you must pass the additional `--user` option. The following output is, like many other listings in this chapter, heavily shortened for reasons of space (on Fedora, the command lists about 60 services!):

```
user$ systemctl --user list-units --type=service
UNIT                  LOAD   ACTIVE SUB     DESCRIPTION
at-spi-dbus-bus       loaded active running Accessibility ...
dbus...               loaded active running D-Bus User Message Bus
dconf.service         loaded active running User preferences database
evolution-...         loaded active running Evolution address ...
```

```
gnome-session-manager... loaded active running GNOME Session Manager
gnome-session-monitor... loaded active running Monitor Session leader
...
gnome-terminal-server    loaded active running GNOME Terminal Server
gvfs-...                 loaded active running Virtual filesystem ...
...
```

systemd searches the following directories for service files:

```
/lib/systemd/user
/etc/systemd/user
~/.local/share/systemd/user
~/.config/share/systemd/user
```

loginctl

Note that systemd runs at the user level, but not at the session level. The systemd process is started when the user in question logs in for the first time and is terminated with the user's last active session. This also applies if a user is logged in multiple times and therefore multiple sessions are active.

An overview of all sessions controlled by systemd is provided by the `loginctl` command:

```
user$ loginctl
SESSION     UID   USER     SEAT   TTY
     c1      42   gdm      seat0  /dev/tty1
      2    1000   kofler   seat0  /dev/tty2
      4    1000   kofler
```

You can use `loginctl show-session` to determine details about a session, including whether the session uses Xorg, Wayland, or just a TTY interface, as is the case with SSH logins:

```
root# loginctl show-session -p Type 2
Type=wayland
```

20.1.5 Additional Functions

Standardization of configuration files

In the course of the changeover to systemd, Lennart Poettering has proposed also standardizing various configuration files, which in the past were stored by almost every distribution in a different location and for which different syntax rules also partly applied. By now, most systemd-based distributions use the configuration files listed in Table 20.1. For details on some of these files and their associated configuration tools, see Chapter 14.

Function	Previous Location	New Location	Command
Host name	/etc/sysconfig/network	/etc/hostname	hostnamectl
Server ID	/var/lib/dbus/machine-id	/etc/machine-id	–
Language	/etc/sysconfig/i18n	/etc/locale.conf	localectl
Keyboard	/etc/sysconfig/keyboard	/etc/vconsole.conf	localectl
Time zone	/etc/sysconfig/clock	/etc/localtime	timedatectl
Modules	/etc/modules	/etc/modules-load.d/*	–
Distribution	/etc/xxx-release	/etc/os-release	–

Table 20.1 Changed Configuration Files

Journal systemd includes a syslog-compatible logging system: the journal. The logging files are stored in a binary format and are thus protected against subsequent changes (see Chapter 14, Section 14.13).

Timer The ystem timer function can take care of the regular execution of processes, similar to Cron. This Cron variant, which is currently still rarely used, is described in more detail in Chapter 10, Section 10.7.

20.1.6 Compatibility

systemd is compatible with the conventional Init-V system. Init scripts located in the /etc/init.d/ directory can thus be started or terminated as before. However, the smaller the number of conventional Init-V scripts in /etc/init.d, the better the respective distribution has been optimized for systemd.

In some distributions, especially Debian and all derived distributions, both Init-V and systemd configuration files are installed redundantly for important services. The systemd files have priority in this case, while the Init-V files are only considered if systemd is *not* used. This is the case, for example, with the Debian alternative Devuan, which does not use systemd.

Services that are started by systemd via Init-V scripts are marked by `systemctl` with the `LSB` label (for *Linux Standard Base*). You can therefore quickly determine the services in question by using `systemctl | grep LSB`. In

most modern distributions, there are no active Init-V scripts anymore; that is, the command returns an empty result.

service command

The `service` command, which was widely used in the past to control init functions, also works with systemd. For example, if you run `service httpd stop`, `service` detects that the web server is controlled by systemd and runs `systemctl stop httpd.service`.

20.1.7 Documentation

systemd is excellently documented. All functions, strategies and advantages of systemd are completely described in various manual pages (especially `man systemd`) and on the website of systemd developer Lennart Poettering:

- *https://systemd.io/*
- *http://0pointer.de/blog/projects/systemd-docs.html*

If you prefer short and concise texts, the following two pages will be useful. They summarize the most important changes compared to Init-V and answer some FAQs:

- *https://fedoraproject.org/wiki/SysVinit_to_Systemd_Cheatsheet*
- *https://www.freedesktop.org/wiki/Software/systemd/FrequentlyAskedQuestions*

20.2 Custom systemd Services

For network services that are available in ready-made packages, you do not need to worry about systemd configuration: the required files are supplied by all distributions. On Fedora, RHEL, and SUSE, you have to start the service using `systemctl enable --now` and enable it permanently. On Debian, Raspberry Pi OS, and Ubuntu, even this step is omitted.

How do you anchor a custom service in systemd? This section explains the syntax of custom systemd service files and presents two usage examples.

20.2.1 Custom systemd Configuration File

To set up a systemd service, you must create a `*.service` file in the `/etc/systemd/system` directory. Again, it's convenient if you use a service file from another comparable service as a guide. The default service files are located in the `/lib/systemd/system` or `/usr/lib/systemd/system` directories, depending on the distribution.

Syntax rules

A minimal service file looks as follows:

```
[Unit]
Description=Foo
[Service]
ExecStart=/usr/sbin/foo-daemon
[Install]
WantedBy=multi-user.target
```

This means that the `Foo` service is supposed to be started as the `foo-daemon` background process. An explicit stop command is missing; therefore systemd will first send the `SIGTERM` signal to terminate and, if that doesn't help, also `SIGKILL` to the process.

A command name with a complete path must be passed to `ExecStart`. In the same way, you can use `ExecStop` to specify a second command to be executed when the background process is to be terminated.

You can distribute extensive commands across several lines with the `\` character.

Within the service file, the `%n` abbreviation is replaced with the name of the service.

Different types of service

Usually you can't specify multiple commands using `ExecStart` or `ExecStop`. If you want to do this, you must specify `Type=oneshot` in the `[Service]` block. Then any number of `ExecStart` or `ExecStop` statements is allowed, and these will be executed in sequence.

If you want to explicitly switch between two states (`start`/`stop`) in such a configuration, then you must also use the `RemainAfterExit` keyword as in Section 20.2.3. The option causes systemd to remember the currently activated state. Without this option, systemd believes that the action has ended with the end of the last `StartExec` command after `systemctl start name` and immediately sets the status back to `stop`. An explicit execution of `systemctl stop` will then have no effect.

For the example in Section 20.2.1, `Type=simple` was implicitly valid. In this case, systemd assumes that the command specified with `ExecStart` is the service to be started. Once this program ends, systemd considers the service to have been properly terminated.

The third important type for services is `Type=fork`. You choose this variant if the command specified with `ExecStart` starts another background program (i.e., a daemon). In this case, you should use `PIDFile` to specify a file where the process number is stored. This allows systemd to monitor the background process or stop it if necessary.

Links

For more information on writing your own systemd service files, see `man systemd.service`, `man systemd.exec`, and the following pages:

- *https://fedoraproject.org/wiki/Packaging:Systemd*
- *https://wiki.archlinux.org/title/Systemd*
- *http://0pointer.de/blog/projects/systemd-for-admins-3.html*

20.2.2 Example 1: Setting up Docker Containers as a Service

The following example shows how you can run a MySQL server as a service in a Docker container. This is useful if the automation provided by Docker (`restart always`) isn't flexible enough for the task and you want to control the container in compliance with other services as needed using `systemctl start`, `stop`, and so on.

systemd service

The `Unit` section of the service file specifies that, as a prerequisite for the service, Docker itself must first be started.

The three `ExecStartPre` statements ensure that the container named `mysql-db-book` is stopped and deleted and that the latest version of the MySQL image is downloaded. The minus signs in front of the `stop` and `rm` commands mean that when these commands are executed, an error will simply be ignored. (If the image has already been stopped and/or deleted before, then this isn't a cause for concern.) The `ExecStart` statement that runs across several lines finally starts the container, using the `/home/kofler/docker-mysql-volume` volume directory and host port 13306. `ExecStop` specifies what needs to be done to stop the service. `RemainAfterExit=true` means that systemd memorizes the currently activated state. Without this option, systemd would assume that the service has already ended after the successful execution of `docker run`. But actually the container continues to run in the background:

```
# File /etc/systemd/system/docker-mysql.service
[Unit]
Description=starts MySQL server as Docker container
After=docker.service
Requires=docker.service
# max. 5 start attempts within 500 seconds
StartLimitIntervalSec=500
StartLimitBurst=5

[Service]
RemainAfterExit=true
ExecStartPre=-/usr/bin/docker stop mysql-container
```

```
ExecStartPre=-/usr/bin/docker rm mysql-container
ExecStartPre=/usr/bin/docker pull mysql
ExecStart=/usr/bin/docker run -d \
    --restart unless-stopped \
    -v /home/kofler/docker-mysql-volume:/var/lib/mysql \
    -p 13306:3306 --name mysql-container mysql
ExecStop=/usr/bin/docker stop mysql-container
Restart=on-failure
RestartSec=5s

[Install]
WantedBy=multi-user.target
```

Start service and activate permanently

For efficiency reasons, systemd keeps the contents of important configuration files in a cache. After configuration changes, you must therefore prompt systemd to reread the configuration files:

```
root# systemctl daemon-reload
```

To start masquerading immediately and in the future every time the multiuser target is reached, the following command is required:

```
root# systemctl enable --now docker-mysql
```

20.2.3 Example 2: Logging the Computer Startup and Shutdown

The goal of the second example is to log the time of each computer startup and shutdown in a separate logging file. This task is performed by the following service file:

```
# file /etc/systemd/system/shutdownlog.service
[Unit]
Description=Log shutdown

[Service]
Type=oneshot
RemainAfterExit=true
ExecStart=/bin/sh -c 'echo "on:  $(date)" >> /var/log/shutdownlog'
ExecStop=/bin/sh -c  'echo "off: $(date)" >> /var/log/shutdownlog'

[Install]
WantedBy=multi-user.target
```

To activate the new service, run the following commands:

```
root# systemctl daemon-reload
root# systemctl enable --now shutdownlog
```

20.3 Shutdown, Reboot, and Halt

The init system is not only responsible for the initialization of a Linux system, but also for the correct shutdown. This elementary operation can be triggered by a menu command on desktop systems. For a server or in a virtual machine without a graphical user interface, you must use the `shutdown`, `reboot`, and `halt` commands instead. The three commands then forward the restart request to the active init system.

shutdown

As the name suggests, `shutdown` shuts down the computer. A time must be specified—either absolutely in the `hh:mm` format or relatively (`+m`, i.e., in `m` minutes) or immediately (`now`). Optionally, you can also specify a message. This message is visible to all users working in a text console or via SSH:

```
root# shutdown -r 23:00 "Reboot for maintenance".
```

The init system then notifies all programs that they will be stopped shortly. Programs have a few seconds to save backups of open files, complete or revoke database transactions, and so on. All programs that are still running after a certain time (usually 20 or 30 seconds) are then forcibly terminated (`kill -9`).

shutdown options

`shutdown` can be supplemented by some options: `-r` causes the computer to be restarted after shutdown (*reboot*). `-f` causes the file system to be checked on the next reboot. `shutdown` creates the `/forcefsck` file for this purpose.

With `-c` (*cancel*), you can revoke a shutdown that is planned for the future if it hasn't started yet.

The additional `-p` option causes the computer to be explicitly switched off after a shutdown. On "real" computers, the option is mostly superfluous. However, virtualization systems often recognize the shutdown of a virtual machine only if the option is specified.

halt and reboot

`halt` and `reboot` are variants of `shutdown`:

- `halt` corresponds to `shutdown now`. In virtual machines, a complete shutdown often requires `halt -p` (discussed previously).
- `reboot` corresponds to `shutdown -r now`.

In distributions with systemd, `shutdown`, `halt`, and `reboot` are implemented as links to `/bin/systemctl`. `systemctl` analyzes how it was started and then initiates the appropriate actions.

Forcing a shutdown/reboot

On my notebook computer, the hibernation mode (suspend mode) doesn't always function properly; errors sometimes occur during its activation and during the wake-up process.

As if that weren't annoying enough, sometimes an "exciting" follow-up problem occurs: the notebook can no longer be switched off or restarted. When the reboot is initiated in the desktop, the notebook just keeps running without any error message. A `shutdown` or `reboot` command in the terminal at least returns an error message:

```
Failed to start poweroff.target: Transaction is destructive
```

A look at the journal then reveals the following:

```
Requested transaction contradicts existing jobs
```

The solution to this is to use the `-f` (*force*) option—for example, `reboot -f`. Alternatively, `systemctl --force reboot` will also do the trick. In particularly stubborn cases, `--force` must be duplicated (`man systemctl` advises against duplicating, but the only other option in such cases is to disconnect the notebook from the power supply by pressing and holding the power button):

```
root# systemctl --force --force reboot
```

20.4 The Traditional Init-V System

The traditional Init-V system is now only used by a very few distributions to perform system initialization and start network services. However, even in current distributions there are occasional services that are still controlled by conventional Init-V scripts. Although systemd takes care of such legacy issues, it is good to know how Init-V is designed and what parts it's composed of.

Procedure

A system startup by the Init-V system is carried out in the following steps (described in shortened form):

- GRUB loads and starts the kernel.
- The kernel starts the `/sbin/init` program.
- `init` analyzes the `/etc/inittab` configuration file.

- `init` executes a script for system initialization.
- `init` executes the `/etc/rc.d/rc` or `/etc/init.d/rc` script. The `rc` script varies considerably from distribution to distribution. It's responsible for starting the script files, which are located in the `/etc/rc<n>.d` or `/etc/init.d/rc<n>.d` directory, depending on the distribution. (`n` represents the runlevel. Details will follow ahead.)
- The script files from `/etc/rc<n>.d` or `/etc/init.d/rc<n>.d` start various system services, especially for the network functions.

20.4.1 Runlevel

For the Init-V system, the concept of a runlevel is of central importance. The allowed runlevels are defined in the `/etc/inittab` file and describe different states that the operating system can have. In the past, there were different numbers or letters to designate the runlevels, depending on the distribution. The following list used to apply to Debian and Raspbian systems. The default runlevel was 2 and was determined by the `initdefault` line in `/etc/inittab`:

- Runlevel S, initialization of the computer immediately after startup
- Runlevel 0, shutdown with stop
- Runlevel 1, single user with network
- Runlevels 2–5, equivalent multiuser operation with network and graphics system
- Runlevel 6, shutdown with reboot

Changing the runlevel

`root` can change the runlevel during operation by using the `init x` command. `x` is a runlevel number or a runlevel letter. For example, for some maintenance tasks, it makes sense to switch to single-user mode. Also `shutdown`, `halt`, `reboot`, or Ctrl+Alt+Del in a text console change the runlevel and lead in this way to a computer restart.

20.4.2 Init-V Scripts

The Init-V system executes so-called Init-V scripts to perform initialization work and to start or stop. Depending on the distribution, these are located in the `/etc/init.d` directory or in `/etc/rc.d/init.d`. To achieve greater compatibility between distributions, links usually ensure that both paths are valid.

The basic idea of the Init-V system is that to activate a runlevel, all scripts required for it are executed with the `start` parameter. When shutting down the computer or switching to a runlevel with fewer functions, the corresponding scripts are executed again, but then with the `stop` parameter.

Init-V script parameters

Most Init-V scripts accept the following parameters:

- `start` starts the relevant function.
- `stop` terminates the function.
- `status` shows short information about whether the function is active or not.
- `reload` is useful if changed configuration files are to be reloaded without stopping the daemon completely. However, not all services provide this option. They may need to be restarted.
- `restart` causes the daemon to be stopped completely and then restarted. Any existing connections to clients will be lost. For database servers, you also lose the cache contents, which causes the database to execute queries much slower for a while.

Starting/stopping services manually

Init-V scripts can be executed not only by the Init-V system, but also manually—for example, to restart a service:

```
root# /etc/init.d/sshd restart
```

Instead of the `systemctl` systemd command, distributions based on Init-V often provide the `service` command. It has the advantage that it also works on most distributions with systemd and is a common denominator of many Linux distributions:

```
root# service sshd restart
```

20.4.3 Links in the Runlevel Directories

Links in runlevel directories indicate which services are to be started or stopped at which runlevel. For example, if the SSH server is to be started at runlevel 2, then the `/etc/rc2.d` directory contains a link such as `S<nn>sshd` that points to the `/etc/init.d/sshd` script. In this link, *S* stands for *start*. When analyzing all S links, the Init-V system passes the `start` parameter to the script. The `nn` number determines the order in which the scripts are executed.

To stop the SSH server on shutdown, `/etc/rc6.d` contains an analogous link, which is named `K<nn>sshd`. In this case, *K* stands for *Kill* and `stop` is passed as a parameter.

20.5 System Startup on Fedora and RHEL

Figure 20.1 summarizes how Fedora or RHEL boots into the default target or how it shuts down again. Table 20.2 provides an overview of the configuration files.

Function	Configuration Files
Binary	`/usr/lib/systemd/systemd`
systemd directories	`/etc/systemd/*`, `/lib/systemd/*`
systemd default target (link)	`/etc/systemd/system/default.target`
System initialization	`/lib/systemd/system/sysinit.target.wants/*`
Configuration files	`/etc/sysconfig/*`

Table 20.2 Configuration of System Startup on Fedora and RHEL

Figure 20.1 also applies to Debian, openSUSE, and Ubuntu. However, on openSUSE, the systemd files are located in the `/usr/lib/systemd` directory instead of `/lib/systemd`. I describe other Debian-, openSUSE-, and Ubuntu-specific features in the following sections.

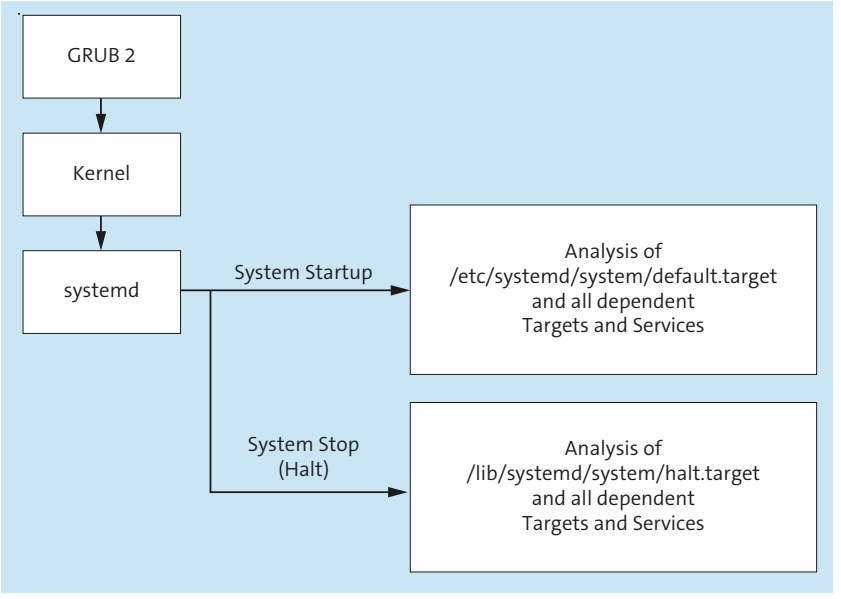


Figure 20.1 Starting and Stopping Debian, Fedora, openSUSE, RHEL, and Ubuntu

Fedora and RHEL have made the move to systemd very consistently. The /etc/init.d/ directory is almost empty, and the /etc/rc.d/rcsysconfig script formerly responsible for system and hardware initialization no longer exists.

Starting the graphic system

The /lib/system/system/gdm.service configuration file is responsible for starting the graphics system.

Configuration

The /etc/rc.d/rc.local file formerly provided for customizing the system startup no longer exists by default on Fedora; on RHEL, it is present but empty. However, if you want to run your own scripts at the end of the system startup, you can still use /etc/rc.d/rc.local for this purpose. To do this, you need to create this file and use chmod +x to make sure that the file is also executable.

20.6 System Startup on Debian, Raspberry Pi OS, and Ubuntu

Debian and Ubuntu also use systemd for the system startup (see Table 1.3). Figure 20.1 therefore also applies to these distributions. However, Init-V compatibility plays a somewhat larger role on Debian or on Ubuntu to this very day. The following listing was created on an Ubuntu server (version 20.04) and shows that isolated services were started by conventional Init-V scripts:

```
root# systemctl
 grep LSB
UNIT              ACTIVE SUB            DESCRIPTION
hddtemp.service   loaded active exited  LSB: disk temperature ...
postgrey.service  loaded active running LSB: Start/stop the postgrey ...
spamass-milter... loaded active running LSB: milter for spamassassin
```

Function	Configuration Files
Binary	/sbin/init (link to /lib/systemd/systemd)
systemd	/etc/systemd/*, /lib/systemd/*
systemd default target (link)	/etc/systemd/system/default.target
System initialization	/lib/systemd/system/sysinit.target*, /etc/init.d/rcS, /etc/rcS.d/*
Conventional init scripts	/etc/init.d/*

Table 20.3 Configuring System Startup in Debian and Ubuntu

Function	Configuration Files
Runlevel links	/etc/rc<n>.d/rc<n>.d/*
Configuration files	/etc/default/*

Table 20.3 Configuring System Startup in Debian and Ubuntu (Cont.)

Automatic activation of new services

Debian and Ubuntu are fundamentally different from the world of Fedora and RHEL in one more respect: newly installed (network) services are started immediately. This eliminates the need for the usual `systemctl start` and `systemctl enable` commands. This is convenient on the one hand, but risky on the other. Make sure that the default configuration corresponds to your ideas!

Graphics system

The `/lib/system/system/gdm.service` configuration file is responsible for starting the graphics system. At startup, the `/etc/X11/default-display-manager` file is analyzed, which contains the file name of the desired display manager.

Customizing the init process

If the `/etc/rc.local` script exists and the following conditions are met, it's executed once when `multi-user` or `graphical.target` is reached. The script is therefore suitable to perform simple configuration tasks without having to deal with the systemd world in more detail.

If you want to use the script, you need to keep three things in mind:

- The script must be executable (`chmod +x`).
- It must start with `#!/bin/bash` and end with `exit 0`.
- Once you have set up the script, you need to run `systemctl daemon-reload` once.

20.6.1 Raspberry Pi OS

On the Raspberry Pi 4 and newer models, the second stage bootloader is located in the ROM and therefore does not need to be loaded from the SD card. Depending on the Raspberry Pi model, the `kernel7l.img` variant optimized for new CPU versions or the `kernel8.img` 64-bit variant is used instead of `kernel7.img`.

Raspberry Pi OS is based on Debian and thus also uses systemd. However, there are big differences in the boot process compared to Debian and Ubuntu. GRUB is not responsible for the boot process. Instead, the boot process is specified by the graphics component (the GPU) of the BCM28xx/BCM2711 chip (see Figure 20.2).

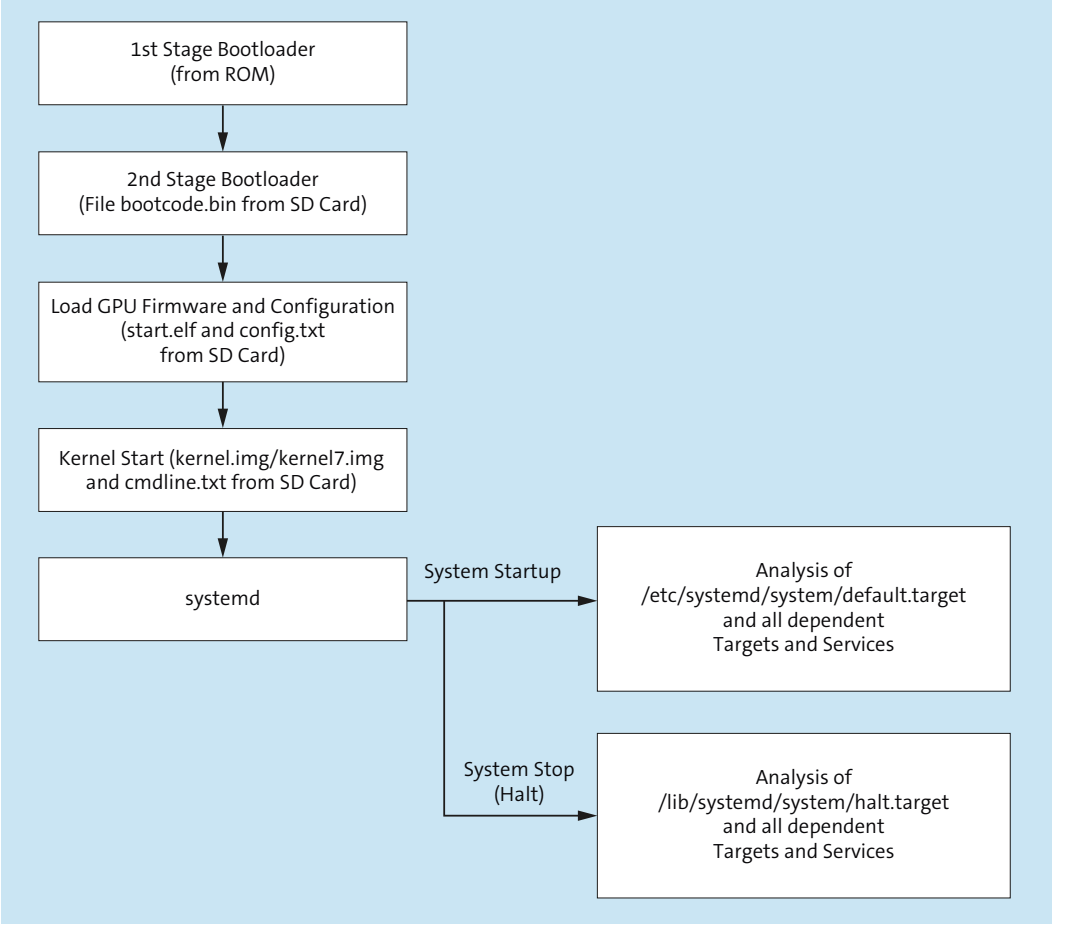


Figure 20.2 Starting and Stopping Raspberry Pi OS

20.7 System Startup on SUSE/openSUSE

SUSE or openSUSE distributions also use systemd for the startup process (see Figure 20.1). Note, however, that on openSUSE the systemd files are located in the `/usr/lib/systemd` directory and that there are also other differences compared to the systemd configuration of other distributions. Table 20.4 lists the configuration files relevant for the system startup.

Function	Configuration Files
systemd	`/etc/systemd/*`, `/usr/lib/systemd/*`
systemd default target (link)	`/etc/systemd/system/default.target`

Table 20.4 Configuration of SUSE System Startup

Function	Configuration Files
system initialization	/usr/lib/systemd/system/sysinit.target*
Configuration files	/etc/sysconfig/*

Table 20.4 Configuration of SUSE System Startup (Cont.)

Starting the graphics system

The /usr/lib/system/system/display-manager.service systemd file is responsible for starting the X graphics system. The DISPLAYMANAGER variable, which is set in /etc/sysconfig/displaymanager, determines which display manager is actually started (e.g., sddm for the Simple Desktop Display Manager).

Chapter 21
Kernel and Modules

This chapter deals with the Linux kernel and its modules. Modules are parts of the kernel that are loaded on demand—for example, when a particular hardware component is addressed for the first time. Let me give you an overview first:

- **Kernel modules**
  Section 21.1 explains why kernel modules are loaded automatically and what you need to do if this automatism fails.
- **Device trees**
  On smartphones and in embedded devices, the requirements for managing kernel modules are quite different from those on PCs. There, device trees for managing kernel modules have become established.
- **Compiling kernel modules yourself**
  If you use special hardware, you may need to compile your own kernel module. How to do this is described in Section 21.2.
- **Compiling the entire kernel**
  Even though it is rarely necessary to recompile the entire kernel, Section 21.3 proves that this is not witchcraft.
- **Kernel update without reboot**
  `kexec` allows you to activate a different kernel without *rebooting* the computer.
- **Kernel live patches**
  It is even more elegant to perform kernel updates while the system is running. Section 21.4 presents mechanisms for activating such live patches.
- **/proc and /sys file system**
  Section 21.5 describes how you can obtain up-to-date information about the kernel from the `/proc` or `/sys` file system.
- **Kernel options and parameters**
  Section 21.6 and Section 21.7 explain how you can pass options to the kernel during computer startup or change kernel parameters during operation.
- **Spectre, Meltdown, and others**
  Finally, Section 21.8 describes the measures the kernel takes to protect you from security vulnerabilities in current CPUs.

This chapter is explicitly intended for advanced Linux users. Beginners are well advised to use only the kernel intended for their distribution! All information in this chapter applies to kernel versions 4.*n*, 5.*n*, and 6.*n*.

21.1 Kernel Modules

The *kernel* is that part of Linux responsible for basic functions such as memory management, process management, access to SSDs and network cards, and so on. The kernel follows a modularized concept:

Initially, when the computer is booted, a base kernel is loaded that contains only those functions required to start the computer. If additional functions are required during operation, such as for special hardware, the necessary code is linked to the kernel as a module. If these additional functions are not needed for a while, the module can be removed from the kernel again. This modularized concept has many advantages:

- Kernel modules can be included as needed. If a certain module is not needed on your computer, memory can be saved in this way, ensuring that the kernel is not larger than absolutely necessary and optimally adapted to your hardware.
- When changing the hardware (e.g., after installing a new network card), no new kernel has to be compiled; only the corresponding module has to be loaded.
- When you develop a kernel module, it is not necessary to constantly reboot the computer because it is sufficient to recompile a module, which can then be tested during operation.

You can find a lot of background information on handling kernel modules in the how-to document available at *https://www.tldp.org/HOWTO/Module-HOWTO*. This document is 15 years old, but the principles of module management have changed little since then.

Loading modules automatically

The kernel's built-in `kmod` component is responsible for ensuring that kernel modules are actually loaded automatically when they are needed. `kmod` is controlled by the `/etc/modprobe.conf` and `/etc/modprobe.d/*.conf` files. These configuration files are described in more detail ahead.

Kernel and modules must match

In the early days of Linux, the kernel and its modules had to fit together exactly: it wasn't possible to load a module compiled for a different, perhaps only slightly modified, kernel version. For this reason, there was and still is a separate module directory named `/lib/modules/n.n` for each kernel version.

In 2006, an attempt was made to store additional information along with a module by using *module versioning*, which provides information on whether interoperability between the module and the kernel is possible even with different version numbers. This meant that modules that did not match the kernel version could often be used.

However, this mechanism only works if module versioning was enabled at compile time and if there have been no interface changes between the kernel and module versions. In practice, this mechanism has not proved particularly successful, which is why its use is uncommon today.

21.1.1 Commands for Module Management

All common distributions are set up to automatically load modules on demand. For example, say that you mount the file system of a USB flash drive in the directory tree. This automatically activates the `vfat` module, which is required to read the file system.

Usually, module management is automatic and transparent, without you having to intervene with the manual module management commands described ahead. Nevertheless, you should know the module commands to be able to load modules manually in case it becomes necessary.

All modules are located in the `/lib/modules/n.n` directory, where `n.n` represents the version of the running kernel. Module files have the `*.ko` or `.ko.xz` file extension if they are compressed with `xz`.

Determining the kernel version

The `uname -r` command returns the version number of the running kernel:

```
user$ uname -r
6.3.1-arch2-1
user$ ls /lib/modules/6.3.1-arch2-1
extramodules
modules.alias
modules.builtin
modules.builtin.bin
...
```

Loading a module file modprobe

`modprobe` loads the specified module into the kernel. For this purpose, you need to specify only the module name. In addition, parameters (options) can be passed to the module. If you want to specify hexadecimal values, you must prefix them with `0x`—for example, `option=0xff`:

```
root# modprobe cifs
```

Compared to the low-level `insmod` command, which I won't discuss here, `modprobe` acts in a relatively intelligent manner:

- `modprobe` itself searches for the module file in the branched module directory tree of the currently active kernel. For modules that have already been integrated into the kernel during compilation, `modprobe` ends without an error message.
- `modprobe` also loads any modules that are required as prerequisites for the desired module, if applicable.
- `modprobe` takes into account all module options specified in `/etc/modprobe*`.

`modprobe` requires correct module configuration through the `/etc/modprobe*` and `/lib/modules/n.n/modules.dep` files.

List of loaded modules

Typically, `lsmod` returns a pretty long list of all currently loaded kernel modules. Note that `lsmod` does not show modules compiled into the kernel, but only those modules that were loaded afterward:

```
user$ lsmod | sort
Module                  Size  Used by
8250_dw                24576  0
ac97_bus               16384  1 snd_soc_core
acpi_pad               24576  0
acpi_thermal_rel       16384  1 int3400_thermal
aesni_intel           401408  12
...
```

The `lspci` command with the `-k` option (for *kernel driver in use*) is very helpful for assigning kernel modules. It lists all hardware components connected via the PCI bus and shows which drivers would be suitable to control the device and which driver is actually being used:

```
user$ lspci -k
00:00.0 Host bridge: Intel Corporation 8th Gen Core Processor
                     Host Bridge/DRAM Registers (rev 07)
        Subsystem: Lenovo 8th Gen Core Processor Host Bridge/
               DRAM Registers
        Kernel driver in use: skl_uncore
00:01.0 PCI bridge: Intel Corporation Xeon E3-1200 v5/E3-1500 v5/
                    6th Gen Core Processor PCIe Controller (x16)
                    (rev 07)
        Kernel driver in use: pcieport
00:02.0 VGA compatible controller: Intel Corporation UHD
                                   Graphics 630 (Mobile)
        Subsystem: Lenovo UHD Graphics 630 (Mobile)
```

```
        Kernel driver in use: i915
        Kernel modules: i915
...
```

Module information

`modinfo` provides a lot of information about a module. The module does not have to be in the kernel:

```
root# modinfo cifs
filename:     /lib/modules/6.3.1-arch2-1/kernel/fs/cifs/cifs.ko
version:      2.42
description:  VFS to access SMB3 servers e.g. Samba, Macs, Azure and
              Windows (and also older servers complying with the SNIA
              CIFS Specification)
license:      GPL
author:       Steve French
...
parm:         cifs_min_rcv: Network buffers in pool. Default: 4
              Range: 1 to 64
parm:         cifs_min_small: Small network buffers in pool.
              Default: 30 ...
...
```

Removing modules

`rmmod` removes the specified module from the kernel and releases the utilized memory. The command can only be executed successfully if the module is not currently being used:

```
root# rmmod cifs
```

21.1.2 Module Configuration

Module management seems to work like magic:

- If you include an additional partition in the file system and a previously unused file system format is used, the module for this file system is automatically loaded.
- If the partition is on a SATA disk, the SATA modules are also activated, if they are not already loaded.
- During the initialization of network functions, the required drivers for your network card are automatically loaded, and so on.

The kernel-integrated `kmod` component is responsible for the automatic loading of kernel modules. Different configuration mechanisms make sure that all this works:

- **Modules required for computer startup**
  Some kernel modules are needed immediately when the computer starts—such as modules for accessing the file system. Unless these modules are integral components of the kernel, they must be passed by GRUB to the kernel in an initrd file when the computer is booted (see Chapter 19, Section 19.1).
- **Modules for basic functions**
  The modules for the basic administration of hardware components (e.g., for the USB system) are loaded directly by various scripts of the init process using `modprobe` statements.
- **Modules for interfaces**
  A number of other modules are then loaded when an interface is used for the first time. However, the problem arises here that there are different modules for some interfaces depending on the hardware used. So if you address the `eth0` interface for the first network card in the computer, the module matching this card must be loaded.

  Because the kernel is not clairvoyant, it needs information about which module is the right one. This information is located in `/etc/modprobe.conf` and in the files of the `/etc/modprobe.d` and `/etc/modules-load.d` directories. There are installation-specific or distribution-specific options as well as statements that define which modules are *not* to be loaded automatically (`blacklist` file). For systemd distributions, this information is analyzed by the service file, `systemd-modules-load.service`.

  The automatic device management by the `udev` system also loads the necessary modules as required. The corresponding rules can be found in `/etc/udev/rules.d`.
- **Modules for USB devices and the like**
  Such hardware components take on a special role. The `/lib/modules/n.n/modules.alias` file contains a reference with identification codes of the components and decides which module gets loaded.
- **Module dependencies**
  A lot of modules are interdependent. For example, the `nfs` module for the NFS file system only works if the `lockd`, `nfs_acl`, and `sunrpc` modules are also loaded. Such module dependencies are listed centrally in the `/lib/modules/n.n/modules.dep` file.

Loading modules at computer startup

Sometimes, independent of the configuration paths summarized here, you may want to have a specific kernel module loaded at computer startup, without relying on any automatisms. The best approach depends on your distribution.

This is especially easy with Debian and Ubuntu. There, the `kmod` command, executed once by systemd, takes care of loading all the modules listed line by line in `/etc/modules`. So you just need to specify the desired module in a new line in `/etc/modules`.

For Fedora and RHEL, you need to add `modprobe modulname` to the `/etc/rc.d/rc.local` init script intended for local customizations. Note, however, that depending on the distribution, this will load the modules only at the end of the init process—that is, at a time when it is already too late for some system and network processes.

21.1.3 modprobe Syntax

The following paragraphs describe the most important keywords for the `modprobe.conf`, `/etc/modprobe.d/*`, and `/lib/modules/n.n/modules.alias` files. Further details are provided by `man modprobe.conf`.

alias

`alias` statements specify which kernel modules are used for which devices. Let's look at an example, in which the `8139too` module is to be used for the `/dev/eth0` device:

```
alias eth0 8139too
```

Access to many hardware components is provided by block-oriented and character-oriented device files in the `/dev` directory. From the kernel's perspective, these device files are not characterized by their name, but by the major and minor device number (see Chapter 9, Section 9.9). Numerous `alias` statements establish the connection between device numbers and modules. The definition of network protocols looks similar: to use a particular protocol, the kernel looks for a protocol family named `net-pf-N`, where `N` is the ID number of the protocol. The following example causes the AppleTalk module to be loaded for protocol family 5:

```
alias net-pf-5 appletalk
```

If you don't need this obsolete protocol and possibly the corresponding module is not installed at all, then the following statement will avoid some annoying error messages:

```
alias net-pf-5 off
```

options

`options` statements specify the options with which a particular module is to be loaded. The following statement causes the `ne` module (for NE-2000 compatible ethernet cards) to be loaded with the `io=0x300` option:

```
options ne io=0x300
```

include

`include` statements load additional configuration files.

install Using `install` statements, you can specify commands that will be executed instead of simply loading the relevant module. Again, you can see an example that has been spread across two lines to save space. If the ALSA module `snd` is required, the following commands are supposed to be executed:

```
install snd modprobe --ignore-install snd $CMDLINE_OPTS && \
  { modprobe -Qb snd-ioctl32 ; : ; }
```

remove We use `remove` to specify commands to be executed when a module is removed.

blacklist `blacklist` causes module-internal alias definitions to be ignored. `blacklist` statements are usually located in the `/etc/modprobe.d/blacklist` file. It contains modules that should *not* be loaded—perhaps because of compatibility problems or because of better alternatives. For example, the following line prevents the `usbmouse` module from being loaded:

```
blacklist usbmouse
```

21.2 Compiling Kernel Modules Yourself

If you use Linux in combination with VirtualBox, want to use NVIDIA's proprietary graphics drivers, or need another hardware-specific kernel module that is missing from your distribution's kernel, you must compile the module to match the running kernel.

Development tools Compiling a module requires the C compiler `gcc` and `make` as well as other basic development tools. Most distributions make things easier by providing ready-made package selections or meta packages that reference all relevant packages (see Table 21.1).

Distribution	Command
Debian, Ubuntu	`apt install build-essential`
Fedora, RHEL	`dnf groupinstall development-tools`
SUSE	`zypper install -t pattern devel_basis`

Table 21.1 Commands for Installing Basic Development Tools

Kernel include files You also need at least the include files (header files) for the current kernel. These files are part of the kernel code. In many distributions (but not in SUSE), the include files and the rest of the code are located in two separate packages. This has the advantage that you do not have to install the huge kernel code if you only need the comparatively small include files. Table 21.2 specifies in which packages the include files of the kernel are located in

the common distributions and where these files are installed. `n.n` is a placeholder for the installed kernel version. You can get this information using the `uname -a` command.

Distribution	Package	Path
Debian	`linux-headers-n.n-amd64`	`/usr/include/linux-headers-n.n`
Fedora/RHEL	`kernel-devel-n.n`	`/lib/modules/n.n/build/include`
SUSE	`kernel-devel`	`/usr/src/linux-n.n/include`
Ubuntu	`linux-headers-n.n-generic`	`/usr/include/linux-headers-n.n`

Table 21.2 Packages with Kernel Header Files

If you compile the kernel yourself (see Section 21.3), the include files that match the kernel automatically end up in the `/lib/modules/n.n/build/include` directory.

Compiling the module

Most programs that require custom kernel modules contain an installation script that takes care of compiling and setting up the module. This applies, for example, to VMware, VirtualBox, NVIDIA's graphics drivers, and others. In some distributions, the process is even automated to the extent that the module is automatically recompiled after each kernel update (see Section 21.2.1).

On the other hand, if you have downloaded the source code for a hardware component that is not yet officially supported, you will have to take care of the compilation process yourself. To do this, you usually need to execute the following commands. Only the final `make` command requires `root` privileges:

```
user$ cd source code directory
user$ make clean
user$ make
root# make install
```

21.2.1 Automating Module Updates

DKMS

Dynamic Kernel Module Support (DKMS) supports the automatic update of self-compiled kernel modules after a kernel update. DKMS consists of several shell scripts and was developed by Dell. The corresponding `dkms` packages are currently available for the Debian, Fedora, and Ubuntu distributions.

To use DKMS, the source code of the module must be installed in a directory in the format /usr/src/NAME-VERSION. The directory must contain the dkms.conf file, which tells DKMS how it must handle the code. The following lines are from the NVIDIA driver for Ubuntu, though I have changed the formatting of the listing to improve readability. In fact, no spaces are allowed before and after the = character:

```
# file /usr/src/nvidia-525.105.17/dkms.conf
PACKAGE_NAME            = "nvidia#"
PACKAGE_VERSION         = "525.105.17"
CLEAN                   = "make clean"
BUILT_MODULE_NAME[0]    = "nvidia"
DEST_MODULE_LOCATION[0] = "/kernel/drivers/char/drm"
PROCS_NUM               = `nproc`
[ $PROCS_NUM -gt 16 ] && PROCS_NUM=16
MAKE[0]                 = "unset ARCH; [ ! -h /usr/bin/cc ] && export
CC=..."
BUILT_MODULE_NAME[1]    = "nvidia-modeset"
DEST_MODULE_LOCATION[1] = "/kernel/drivers/char/drm"
BUILT_MODULE_NAME[2]    = "nvidia-drm"
DEST_MODULE_LOCATION[2] = "/kernel/drivers/char/drm"
...
```

If these requirements are met, you transfer the control of the kernel module to DKMS by using dkms add, then compile the module for the current kernel using dkms build, and install it by using dkms install.

The following examples again refer to the NVIDIA kernel driver. The commands are usually executed automatically after updating the kernel or a driver (dkms autoinstall command). Only if this automatic function does not work do you have to help manually and search for the causes of any errors (sometimes the versions of the kernel and the driver are not compatible with each other):

```
root# dkms add     -m nvidia -v 525.105.17
root# dkms build   -m nvidia -v 525.105.17
root# dkms install -m nvidia -v 525.105.17
```

These commands apply to the currently running kernel. To compile and install kernel modules for a different kernel version, you need to add the -k n.n option to the commands, where n.n is the version number of a kernel installed on your computer. dkms status, or a look in the /var/lib/dkms directory, reveals which kernel modules are currently controlled by DKMS.

DKMS on Debian and Ubuntu

Debian and Ubuntu have a whole set of `xxx-name-dkms` packages that can be used to compile kernel modules for hardware drivers:

```
root# apt-cache --names-only search '.*-dkms' | sort
```

On Debian, some of the packages are in the contrib and nonfree package sources, which you may need to enable upfront. The installation of a kernel module then looks like this, where you simply replace `nvidia` with the name of the relevant driver and `amd64` with your CPU architecture:

```
root# apt update
root# apt install linux-headers-n.n-amd64 nvidia-nnn-dkms
```

As part of the installation, all dependent packages are installed and the kernel module is compiled and set up as a DKMS module. This means that a new version of the module will automatically be created for future kernel updates.

DKMS or module-assistant?

Especially on Debian, the tools from the `module-assistant` package were often used to compile kernel modules in the past. The command of the same name or its short form, `m-a`, still exists, but it must be installed separately. If possible, you should prefer DKMS packages!

akmods

The RPMFusion package source for Fedora uses its own mechanism for compiling kernel modules instead of DKMS. The `akms` command (see *https://rpmfusion.org/Packaging/KernelModules/Akmods*) is called after kernel updates. It compiles the corresponding kernel modules for all RPM source packages located in `/usr/src/akmods` and installs them.

21.3 Configuring and Compiling the Kernel Yourself

The average Linux user does not need to compile his or her own kernel. All current distributions come with a usable standard kernel and an extensive collection of modules. Nevertheless, there may be reasons to recompile the kernel:

- You want to get to know your system better. After all, the idea of this book is to give you a look behind the scenes of Linux.
- You need special functions that are neither integrated into the supplied kernel nor available as a module.
- You want to use a more recent version of the kernel than the one that came with your distribution.

- You want to participate in the kernel development yourself and therefore experiment with the latest developer kernel.
- You want to show off insider knowledge to your friends: "I compiled the latest Linux kernel myself!"

Hurdles

However, there are weighty reasons not to compile your own kernel:

- Many distributions do not use the original kernel as released by Linus Torvalds, but a patched version with various additional features. Of course, each distribution uses different patches.

  In itself, this is a great thing for the user: they get additional functions that the distributor believes are already sufficiently stable. However, if you then download the source code of the original kernel yourself, these patches are missing. Individual functions of your distribution, which previously worked flawlessly, suddenly cause problems or no longer work at all.
- Compiling your own kernel is not difficult. What is difficult, however, is the prior configuration of the compilation process. There are thousands of options to choose from. You can use these options to influence which functions should be integrated directly into the kernel, which should be available as modules, and which should not be available at all.

  If you choose the wrong options—due to a lack of detailed knowledge—the result will be again that individual functions refuse to work, and it is relatively difficult to determine the cause. Especially for Linux beginners, it's practically impossible to correctly guess the appropriate settings for all options.

For these reasons, most distributors refuse to provide any support unless you use the kernel that comes with the distribution. However, don't let these warnings deter you from giving it a try yourself. If you follow the instructions presented in this section, you will be able to boot your computer with both the old and the new kernel afterward—so nothing can go wrong!

Development tools

The same development tools are required to compile the kernel as to compile a single module (see Section 21.2).

Depending on the distribution, additional packages may be required. For example, on Debian and Ubuntu, you need to enable the `deb-src` package sources in `/etc/apt/sources.list` and then run the following commands:

```
root# apt update
root# apt install flex bison
root# apt build-dep linux
```

On Fedora, the following packages are often not installed automatically:

```
root# dnf install dwarves ncurses-devel zstd
```

21.3.1 Basic Principles

Kernel versions

Further development of the Linux kernel has been taking place at the same pace for years now: as soon as Linus Torvalds finds that the kernel version currently under development is sufficiently stable, it is officially released. At the same time, a phase of about two weeks starts during which various developers send Linus Torvalds patches for new functions. After that, it typically takes five to seven weeks to fix all the problems and bugs in the new code and get the next kernel version ready.

Version numbers

The kernel version number consists of three parts: *major release number*, *minor release number*, and *bugfix release number*. When I revised this chapter in mid-May 2023, kernel version 6.3 was the latest one. It had been released by Linus Torvalds on April 23. Until mid-May, there was only one minor update to fix bugs and security issues. So the full version number at this point was 6.3.1.

Long-term releases

Basically, the maintenance of old kernel versions is terminated with the completion of the latest version. Exceptions to this rule are selected kernel versions that have been maintained by the kernel developer community for several years. Currently, the following kernel versions enjoy long-term support: 4.14, 4.19, 5.4, 5.10, 5.15, and 6.1.

For Linux distributors, there are three ways to handle kernel updates:

- On the surface, it would be easiest to always offer the latest kernel version as an update. However, this can entail stability issues (especially if other components, such as the graphics system, are not updated as well), which is why most distributors shy away from this route. The few exceptions include Fedora, as well as rolling release distributions such as Arch Linux or openSUSE Tumbleweed.
- The next best variant is to use the latest long-term Linux version for the distribution. This has the great advantage for the distributor that they do not have to take care of the maintenance of the kernel code themselves.
- The most complex situation is when a distributor does not use a long-term kernel, but maintains the kernel originally supplied with the distribution himself. For this purpose, security updates and, in some cases, additional functions must be "backported" from current kernel code. Particularly noteworthy in this regard is Red Hat, which has spent an entire decade providing old kernel versions with security updates for its enterprise distributions with immense effort.

Statistics The kernel currently consists of about 36 million lines of code. Most of it is written in C, with small parts in Assembler as well as in the Rust language. If you want to know who or which companies contribute to kernel development, just follow the Linux news site at *https://lwn.net*. There you will find a statistical analysis of who has made the most changes for each kernel release, such as at *https://lwn.net/Articles/929582* (for version 6.3).

Links For tips on compiling the kernel, see *https://kernelnewbies.org/FAQ*.

If you're interested in internal technical details, the kernel code documentation files at *https://www.kernel.org/doc/Documentation* are very informative. Especially new features of the kernel are first described there, even before the corresponding `man` pages are updated.

21.3.2 Installing the Kernel Code

The source code for the kernel is usually located in the `/usr/src/linux` directory; only Red Hat and Fedora have different procedures, which are discussed ahead. If this directory is empty, you haven't installed the kernel code. You can then choose to install your distribution's kernel source code or download the latest official kernel code. The first variant usually causes fewer problems, especially for beginners.

> **Is There Enough Space on the Hard Disk?**
>
> The space required for the kernel code is considerable: the compressed source code packages are more than 120 MiB in size. After unpacking, the footprint is about another GiB, and after compiling with the resulting binaries, over 20 GiB! Eventually, you can use `make clean` to delete countless object files, but in the meantime you need a sufficient amount of free space.

Installing the kernel code of the distribution Most distributions have a separate package that contains the kernel source code. Table 21.3 specifies which packages contain the kernel code for some common distributions, where `n.n` is a placeholder for the installed kernel version.

Distribution	Package
Debian, Ubuntu	`linux-source-n.n`
Fedora, Red Hat	`kernel-n.n` (source code package!)
SUSE	`kernel-source`

Table 21.3 Packages with Kernel Source Code

For Debian and Ubuntu, the kernel code is installed as a TAR archive in the `/usr/src` directory. You have to unpack the archive yourself using `tar xjf linux-n.n.tar.bz2`. The `.bz2` identifier indicates that the source code was compressed using the particularly efficient bzip2 method.

Fedora

Fedora is a particularly popular distribution among kernel developers. Before you get started, you must install various developer packages and unlock the current user account for `pesign` (`pesign` is a command for signing UEFI programs):

```
user$ sudo dnf install fedpkg fedora-packager rpmdevtools \
        ncurses-devel pesign openssl
user$ sudo pesign
pesign: Nothing to do.
```

The Fedora developers recommend downloading the source code of the kernel and all Fedora patches from a Git repository via `fedpkg clone -a kernel`. You can find more details on this topic at the following address: *https://fedoraproject.org/wiki/Building_a_custom_kernel*.

Installing the official kernel code

The kernel that comes with the distribution is often already outdated. You can find the current kernel code in the form of compressed TAR archives at *https://www.kernel.org*.

Instead of downloading and using a specific kernel version, it may make more sense to create a clone of the official Git repository for stable kernel versions. `git tag`, in combination with `sort -V` (numerical sort order), determines the 10 most recent versions contained in the code. `git checkout` activates the version you actually want to use (compile) afterward; in this case, that's release candidate 2 of version 6.4, which is still under development:

```
user$ git clone \
  git://git.kernel.org/pub/scm/linux/kernel/git/stable/linux-stable.git
user$ cd linux-stable
user$ git tag -l | sort -V | tail -n 10
v6.3-rc2
v6.3-rc3
v6.3-rc4
v6.3-rc5
v6.3-rc6
v6.3-rc7
v6.3.1
v6.3.2
v6.4-rc1
v6.4-rc2
user$ git checkout v6.4-rc2
```

Initially, `git clone` downloads significantly more code (download size about 4 GiB; space requirement on the SSD about 6 GiB); however, subsequent updates or version changes are easier and faster—assuming a basic knowledge of the `git` command.

Whether a TAR archive or via Git, note that this is the original kernel code as released by Linus Torvalds—that is, without distribution-specific patches. This kernel is often referred to as the *vanilla kernel*.

Variants and Developer Branches

There are countless other Git repositories on the internet with various variants and development branches of the kernel code. In particular, you should take a look at kernel developer Greg Kroah-Hartman's blog (fourth link!):

- *https://git.kernel.org*
- *https://github.com/torvalds/linux*
- *https://git.kernel.org/pub/scm/linux/kernel/git/torvalds/linux.git*
- *http://www.kroah.com/log/blog/2019/06/15/linux-stable-tree-mirror-at-github*

21.3.3 Using Supplied Kernel Configuration Files

The kernel consists of thousands of individual functions or components. For almost all functions, you can specify before compiling whether they should be integrated directly into the kernel, compiled as a module, or not available at all. This process is referred to as *configuring the kernel*.

.config file

The kernel configuration is determined by the `.config` file in the `/usr/src/linux-n.n` directory. This is a text file of nearly 8,000 lines, which specifies whether a function is to be integrated directly into the kernel (`name=y`) or compiled as a module (`name=m`).

Functions that are not required do not appear in the configuration file or only in comment lines. The file can also contain additional settings (`name=value`). The following lines show the beginning of a `.config` file:

```
CONFIG_64BIT=y
CONFIG_X86_64=y
CONFIG_X86=y
CONFIG_INSTRUCTION_DECODER=y
CONFIG_PERF_EVENTS_INTEL_UNCORE=y
CONFIG_OUTPUT_FORMAT="elf64-x86-64"
```

If you do not have a starting point in the manual kernel configuration (see Section 21.3.4), you really need to take care of all kernel options. Especially the first time, it's almost certain that you will overlook something. You can save a lot of time and work by using the kernel configuration file that comes with your distribution as a starting point:

```
user$ cp old-config /path/to/code/linux-n.n/.config
```

Unfortunately, this method has one drawback: if the original kernel code contains different patches than the code being recompiled, the original configuration file also contains options that are not provided in the new code. This can result in problems. As I mentioned earlier, many distributors include various patches in their kernels that are not included in the standard kernel. This is why you need to change to the source code directory afterward and run the following command there:

```
user$ cd /path/to/code/linux-n.n
user$ make oldconfig
```

`make oldconfig` analyzes the existing `.config` file. If there are options missing that are provided by the current kernel code, then corresponding queries are displayed. Usually you can simply answer by pressing Enter; then the default settings suggested by the developers will apply to these options. This is exactly what the following command does—without any query:

```
user$ make olddefconfig
```

Determining the current configuration

Now the question remains where you get the current kernel configuration file for the `cp old-config` command. In almost all distributions, the `/boot` directory contains the configuration file that matches the running kernel—for example, `/boot/config-n.n`. Thus, `cp old-config` becomes, for example:

```
user$ cd /path/to/code/linux-n.n
user$ cp /boot/config-6.2.11-300.fc38.x86_64.config
user$ make oldconfig
```

cloneconfig

The kernel shipped with SUSE uses the `cloneconfig` option (general setup group). This means that `/proc/config.gz` contains the compressed contents of the `.config` file used to compile the currently running kernel. You can use `make cloneconfig` to copy the most recently used configuration into the `.config` file.

21.3.4 Configuring the Kernel Manually

Monolithic or modularized kernel?

In general, you need to decide between two kernel types: monolithic kernels and modularized kernels. *Monolithic kernels* contain all required drivers directly in the kernel and do not support any modules. *Modularized kernels* are capable of accommodating additional modules beyond the built-in drivers during operation. A modularized kernel is the better choice in almost all cases.

Component selection

For most components, you have a choice of three options: **Yes**, **Module**, and **No**. **Yes** means that this component is integrated directly into the kernel. **Module** means that this component is compiled as a module (only useful with a modularized kernel). **No** means that the component will not be compiled at all. There are also a number of functions that cannot be provided as modules; there, the choice is reduced to **Yes** or **No**.

Configuration strategies

The usual approach is to include only relatively few elementary functions in the modularized kernel and to make all other functions available as modules. This has the following advantage: the kernel itself is relatively small, and modules are only reloaded as needed.

An alternative strategy is to optimize a monolithic kernel as precisely as possible for your own hardware and software requirements. You integrate all functions that are to be used directly into the kernel.

Regarding all other components, you should opt for **No**. In general, a monolithic kernel will always be slightly larger than a modularized kernel. In return, it works without the dynamic module management, and the computer startup succeeds without an initrd file. The disadvantage is obvious: if you need a certain function later, you have to recompile the kernel, and only real Linux professionals can estimate which features they will use.

21.3.5 Tools for a Manual Kernel Configuration

To change individual settings in a way that deviates from the current configuration, you can edit `.config` manually. However, this is prone to errors and requires a good knowledge of the names of the various options. It is therefore better to start a special configuration program using `make xxxconfig`. There are different variants available, which you can start by using one of the following `make` commands:

```
user$ cd /usr/src/linux-n.n
user$ make menuconfig          (dialog-guided configuration in text mode)
user$ make nconfig             (dialog-guided configuration in text mode)
user$ make localmodconfig      (automatic configuration for the current hardware)
```

make menuconfig and make nconfig

`make menuconfig` and `make nconfig` require that you have previously installed the `ncurses-devel` and `libncurses5-dev` packages, respectively. The configuration is also done in text mode. The big advantage compared to `make config` is that the setting of the countless options is structured by nested dialogs (see Figure 21.1). The designs of the dialogs and the keyboard shortcuts vary a little, but the functionality of both variants is pretty similar.

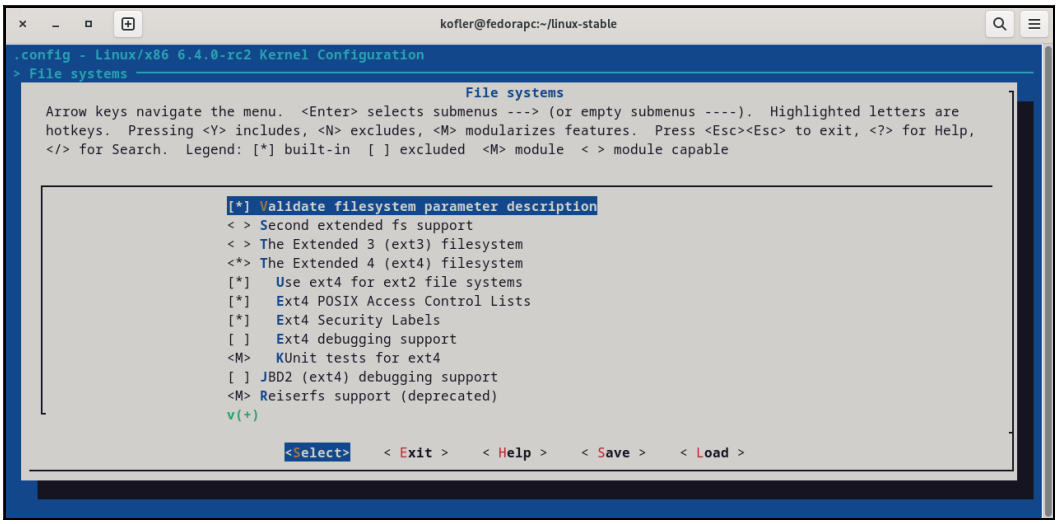


Figure 21.1 Kernel Configuration Using "make menuconfig"

make localmodconfig

`make localmodconfig` is an interesting compilation variant for those who are in a rush. Only those modules are compiled that are actually used in the currently running kernel. This has advantages and disadvantages. The obvious advantage is that only the part of the kernel code that is actually needed is translated. This can reduce the translation time to one-third!

However, the kernel compiled in this way may not run on another machine if drivers relevant to its hardware components are missing. In addition, the reloading of a module that was not active at compile time will also fail. Thus, the kernel is only suitable for testing purposes, but not for long-term use.

Implementing only a few changes

If you only want to change individual options on a given kernel configuration, it is best to avoid `make xxxconfig` altogether. Instead, you should specify the settings you want to implement in the separate `config-local` file. The options selected here take precedence over `.config`.

21.3.6 Compiling and Installing the Kernel

Compiling the kernel and modules

After you've spent some time configuring the kernel, the computer must start working. The following commands will keep a modern computer (SSD, six CPU cores with hyperthreading) busy for about 20 to 30 minutes. The `-j N` option specifies how many processes `make` should start in parallel:

```
user$ cd /path/to/code/linux-n.n
user$ make -j 12 all                    (compile everything; 12 virtual CPU cores)
```

The result at the end of this process is the `bzImage` file in the `/path/to/code/linux-n.n/arch/x86_64/boot` directory. The size of the file is usually between 5 and 10 MiB and depends on how many functions are directly included in the kernel and how many are compiled as modules or not compiled at all.

Error during Compilation

If an error occurs during compilation, you should, of course, try to get to the bottom of it. If the problem occurs with a function that is not important to you, you can change the configuration so that the affected function is not compiled.

Hardcore Linux geeks can simply call `make` with the additional `-k` option—for example, `make -k all`. This option causes errors to be ignored so that `make` simply proceeds to compile the next file. If you're lucky, the compilation issue concerns a module that is unimportant to you, and it isn't available.

Installing modules

`make modules_install` copies the module files to where the module management commands (such as `insmod`) expect them—namely, to the `/lib/modules/n.n` directory. Here, `n.n` represents the version number of the kernel just compiled:

```
user$ sudo make modules_install         (install modules including debug information)
```

By default, the newly created kernel modules contain a lot of debugging information. This is why they are more than 10 times bigger than "ordinary" modules and result in huge initrd files (in my tests, 400 MiB instead of 38 MiB!). Unless you plan to go searching for kernel errors, you should remove the debugging information from the modules. (In the kernel documentation, this process is referred to as *stripping*.) Instead of the ealier command, you must run the following variant:

```
user$ sudo make INSTALL_MOD_STRIP=1 modules_install   (install modules only)
```

Installing the kernel

The freshly created new kernel is not yet active, of course. Up to this point, only a lot of new files have been created, but nothing else! The new kernel cannot be activated until the next time you start Linux, and even then only if you copy the kernel to the `/boot` directory and create a matching initrd file. It is also common to store the `.config` file used during compilation as `config-n.n` in the `/boot` directory. This way, you can document how your kernel was compiled.

Fortunately, `make install` takes care of all the points just summarized:

```
user$ sudo make install                          (install kernel files)
```

If your system requires kernel modules whose code is outside the official kernel code (typically the NVIDIA driver), these modules must also be recompiled and accounted for. In many distributions, this is done by means of DKMS. Note that compiling the NVIDIA driver often fails with new kernel versions or requires the very latest version of the NVIDIA driver!

Finally, you need to update the GRUB configuration:

```
root# update-grub                                (Debian and Ubuntu)
root# grub2-mkconfig -o /boot/grub2/grub.cfg     (Fedora, RHEL, SUSE)
```

Restart and cleanup

You'll notice whether everything has worked when you perform a reboot:

```
user$ uname -a
Linux fedorapc 6.4.0-rc2 ...
```

If for some reason the new kernel doesn't work, you should simply boot the computer with the previous kernel and make another attempt to properly configure and recompile the kernel. If, on the other hand, the new kernel runs satisfactorily, you should clean up the object files of the compiler that are now no longer needed. You will regain about 15 GiB of space on your SSD this way:

```
root# cd /path/to/code/linux-n.n
root# make clean
```

21.4 Kernel Live Patches

Most updates of a Linux system can be done while the system is running. Updated network services will need to be restarted afterward, but there is no need to reboot the entire computer. The exception to this rule is the kernel. For security updates to take effect in the kernel, you must install a new

kernel and new modules, and then reboot the entire machine or, via `kexec`, the kernel and all services.

On notebook computers, which are usually turned on and off every day, it makes no difference. But for servers that are supposed to be constantly available without interruption, a restart is mostly undesirable. And even administrators who automate updates are usually reluctant to automate the necessary reboots as well. The risk is too great that something will go wrong (usually in the middle of the night) and then no one will have time to fix the problem.

Ksplice

The first solution to this problem was the *Ksplice* function. Many updates make it possible to deactivate the relevant kernel function during operation and replace it with new code. The nontrivial technical background of the process is described on the following pages:

- *https://ksplice.oracle.com*
- *https://lwn.net/Articles/340477*
- *https://en.wikipedia.org/wiki/Ksplice*

In mid-2011, Oracle acquired Ksplice. Kernel updates performed using the Ksplice function were a quite significant differentiator compared to other Enterprise Linux versions for several years. However, Oracle only offers Ksplice to paying customers. Ksplice is available as open-source code but is not part of the official Linux kernel.

kPatch and kGraft

Red Hat and SUSE did not want to back down in this respect, of course, and developed comparable update mechanisms named *kPatch* and *kGraft*. These two mechanisms have been available in the enterprise versions of Red Hat and SUSE since 2014. The functions required for kPatch and kGraft (the greatest common denominator, so to speak) were included in the official Linux kernel by Linus Torvalds in 2015. You can tell if your kernel contains the functions by the presence of the `/sys/kernel/livepatch` directory. However, Red Hat and SUSE also provide suitable live kernel patches only to paying customers. For more information, visit the following pages:

- *https://github.com/dynup/kpatch*
- *https://en.wikipedia.org/wiki/KGraft*

Ubuntu

Canonical entered the live patching business last. Since late 2016, Canonical has offered live patches to its commercial customers for critical security issues in Ubuntu LTS versions, first as part of the landscape services, and now under the Ubuntu Pro name. The underlying `canonical-livepatch` program makes use of the live patch infrastructure in the kernel mentioned earlier.

Fortunately, Canonical also makes the live patches available to noncommercial users to a limited extent. Provided you have a free Ubuntu One account, you will receive a token to add five computers to the Ubuntu Pro program:

```
user$ sudo apt install pro
user$ sudo pro attach <token>
...

This machine is now attached to 'Ubuntu Pro - free personal
subscription'

SERVICE     ENTITLED  STATUS    DESCRIPTION
esm-apps         yes  enabled   Expanded Security Maintenance for
                                 Applications
esm-infra        yes  enabled   Expanded Security Maintenance for
                                 Infrastructure
fips             yes  disabled  NIST-certified core packages
fips-updates     yes  disabled  NIST-certified core packages with
                                 priority ...
livepatch        yes  enabled   Canonical Livepatch service
usg              yes  disabled  Security compliance and audit tools
```

As part of the Ubuntu Pro activation, the live patch service is also set up (see the penultimate line in the preceding listing). The required software is only available as a Snap package. You can determine the live patch status by using `canonical-livepatch status`:

```
root# canonical-livepatch status
last check: 24 minutes ago
kernel: 5.4.0-74.83-generic
server check-in: succeeded
patch state: OK, all applicable livepatch modules inserted
tier: updates (Free usage; This machine beta tests new patches.)
```

Pay Attention to the Small Print!

The status message contains a hint that is easy to miss: *This machine beta tests new patches*. Specifically, this means that as a nonpaying Ubuntu Pro user, you are a beta tester for kernel updates! Canonical delivers kernel patches to nonpaying customers first. If they don't cause any problems, the same updates will be made available to paying customers later.

Basically, this approach is understandable. I use the free Ubuntu Pro offering on several machines/servers and have been very happy with it so far.

However, Canonical's nontransparent information policy is annoying. Although I've been searching for it for a while, I haven't come across any official document on the internet that clearly indicates the beta nature of the kernel updates when they are used for free.

If you want detailed information about fixed vulnerabilities, you must additionally specify the `--verbose` option (the following results are from another computer that has not been rebooted for a while):

```
root# canonical-livepatch status --verbose
  * cve-2023-0461
    It was discovered that the Upper Level Protocol (ULP)
    subsystem in the Linux kernel did not properly handle sockets
    entering the LISTEN state in certain protocols, leading to a
    use-after-free vulnerability. A local attacker could use this
    to cause a denial of service (system crash) or possibly execute
    arbitrary code.
  * cve-2023-1281
    It was discovered that the Traffic-Control Index (TCINDEX)
    implementation in the Linux kernel contained a use-after-free
    vulnerability. A local attacker could use this to cause a denial
    of service (system crash) or possibly execute arbitrary code.
```

`dmesg | grep livepatch` returns messages about recent live patches:

```
root# dmesg | grep 'livepatch:'
livepatch: enabling patch 'lkp_Ubuntu_4_15_0_200_211_generic_94'
livepatch: 'lkp_Ubuntu_4_15_0_200_211_generic_94': starting patching
           transition
livepatch: 'lkp_Ubuntu_4_15_0_200_211_generic_94': patching complete
```

Live Patches for Emergencies Only

Changing the code of the running kernel is a delicate operation (if you're fond of dramatic comparisons, it's like open heart surgery). For this reason, this mechanism is currently used by all distributors only for dangerous vulnerabilities in the kernel.

For other corrections, a new kernel is installed as before, but the activation of the changes made there via live patches is waived. Thus, even if live patches are enabled, your distribution may report that a reboot is required after installing (kernel) updates. Don't let this put you off.

21.5 The /proc and /sys Directories

The /proc and /sys directories are mounted to the file system during system startup. They are used to make information about the kernel, running processes, loaded modules, and many other parameters visible in a transparent way.

Internally, the /proc and /sys directories are implemented as virtual file systems. Thus, they do not contain real files and therefore do not take up any space on the hard disk. This also applies to the seemingly very large /proc/kcore file, which maps the memory.

Most of the /proc and /sys files are in text format.

To read the files, you may have to use cat instead of less, because some less versions cannot handle virtual files.

The /proc directory provides a lot of internal kernel information as well as data about all currently running processes (see Table 21.4). Among other things, each process is assigned its own subdirectory there. Inside the process directory, there are then some files with various administration data (such as the command line used for startup). This data is analyzed by various commands for process management (e.g., top, ps, and so on).

File	Meaning
/proc/N/*	Information about the process with PID=N
/proc/asound	ALSA (Advanced Linux Sound Architecture)
/proc/bus/usb/*	USB information
/proc/bus/pccard/*	PCMCIA information
/proc/bus/pci/*	PCI information
/proc/cmdline	Parameters passed to the kernel
/proc/config.gz	Kernel configuration file (SUSE)
/proc/cpuinfo	CPU information
/proc/devices	Numbers of active devices
/proc/fb	Information on frame buffers
/proc/filesystems	File system drivers included in the kernel
/proc/interrupts	Usage of the interrupts
/proc/lvm/*	Usage of the Logical Volume Manager

Table 21.4 Important /proc Files

File	Meaning
/proc/mdstat	RAID state
/proc/modules	Active modules
/proc/mounts	Active file systems
/proc/net/*	Network state and usage
/proc/partitions	Hard disk partitions
/proc/scsi/*	SCSI/SATA devices and controllers
/proc/sys/*	System and kernel information
/proc/uptime	Time in seconds since computer startup
/proc/version	Kernel version

Table 21.4 Important /proc Files (Cont.)

The /sys directory contains some of the same information as /proc, but the data is organized more systematically (see Table 21.5). The purpose of the /sys directory is to map the relationship between the kernel and the hardware.

File	Meaning
/sys/block/*	Information about all block devices (hard disks, etc.)
/sys/bus/*	Information about all bus systems (PCI, SCSI, USB, etc.)
/sys/class/*	Information about device classes (Bluetooth, graphics, etc.)
/sys/devices/*	Information about connected hardware components
/sys/firmware/*	Information about hardware drivers and firmware
/sys/kernel/*	Information about the kernel
/sys/module/*	Information about loaded modules
/sys/power/*	Information about power management

Table 21.5 Important /sys Files

21.6 Kernel Boot Options

The kernel does not always have to be recompiled if a detail in the kernel is to be changed! There are two ways to affect the kernel without recompiling:

- First, you can use the boot loader to pass parameters to the kernel during system startup. This mechanism is the subject of this section.
- Alternately, you can change a number of kernel functions *dynamically*—that is, during operation. This type of intervention is particularly common for controlling network functions and is described in Section 21.7.

When configuring GRUB, you can specify kernel boot options in the `linux` line or in the `/etc/default/grub` file. You can also type such options interactively when starting a Linux installation program or when starting GRUB from the keyboard (see Chapter 19, Section 19.2). The syntax for specifying options looks as follows: **GRUB**

```
linux /boot/vmlinux-n.n optionA=parameter optionB=
parameter1,parameter2
```

The parameters for an option must be specified without spaces. Multiple options must be separated by spaces, not commas. Hexadecimal addresses are specified in the format `0x1234`. Without a preceding `0x`, the number is interpreted as a decimal number.

Kernel boot options often help to work around hardware problems. For example, if the Linux kernel doesn't recognize how much RAM your machine has due to a faulty BIOS, you can specify the correct value by using the `mem=` parameter.

Note that the parameters specified at Linux startup only affect the drivers integrated into the kernel! Parameters for kernel modules, on the other hand, must be specified in the `/etc/modprobe.conf` file or in the `/etc/modprobe.d` or `/etc/modules-load.d` directory.

This section describes only the most important kernel boot options. For more information, please refer to `man bootparam` and check the following pages:

- *https://tldp.org/HOWTO/BootPrompt-HOWTO.html*
- *https://www.kernel.org/doc/Documentation/admin-guide/kernel-parameters.txt*

21.6.1 Important Kernel Boot Options

The following list contains key kernal boot options:

- **`root=/dev/sdb3`**
  The `root` option specifies that after the kernel has been loaded, the third primary partition of the second SATA drive is to be used as the system partition for the root file system. Other drives and partitions can of course also be specified in the same way.

If the partition is labeled, the system partition can also be specified using the `root=LABEL=xxx` format. Fedora and Red Hat in particular make use of this option. The / character is usually used as the name for the system partition. For `ext` partitions, you must determine the partition name via `e2label` or change it using `tune2fs`.

Another variant is to specify the system partition by `root=UUID=nnn`, where `nnn` is the UUID of the hard disk partition. You can determine this identification number via `/lib/udev/vol_id partition`.

- **`ro`**
  The `ro` option specifies that the file system should be mounted *read-only* for now. This is useful (in combination with one of the following two options) when a corrupt file system needs to be repaired manually.
- **`init`**
  After starting the kernel, most distributions start systemd (see Chapter 20). If you do not want this, you can specify another program with the `init` option.

  By using `init=/bin/sh`, for example, you can make sure that a shell is started. The option can help Linux professionals get a Linux system up and running again if something went wrong during init configuration. Note that the root file system is only available in read-only mode. (You can change this using `mount -o remount`; see Chapter 18, Section 18.8.) Also note that the `PATH` variable is still empty.
- **`single` or `emergency`**
  If you use one of these options, the computer starts in single-user mode (Init-V) or rescue mode (systemd). Strictly speaking, these options are not analyzed by the kernel but are passed on to the first program started by the kernel in the same way as all unknown options (see Chapter 20).
- **`initrd=name`**
  `initrd` specifies the name of the initial RAM disk file to be loaded. If you do *not* want to use an initrd file, you need to specify `initrd=` or `noinitrd`.
- **`ipv6.disable=1`**
  This option deactivates all IPv6 functions of the kernel.
- **`reserve=0x300,0x20`**
  This option specifies that the 32 bytes (hexadecimal 0x20) between 0x300 and 0x31F must not be accessed by any hardware driver to search for any components in them. The option is necessary for some components that are allergic to such tests. It usually occurs in combination with a second option that specifies the exact address of the component that claims this memory area.

- **pci=bios|nobios|nommconf**
  This option controls how the hardware detection of PCI components is carried out. If problems occur, sometimes `pci=bios` or `pci=nommconf` helps.
- **quiet**
  This option causes no messages to be displayed on the screen during kernel startup.
- **video=1024x768**
  This option can be used to set the preferred graphics resolution via Kernel Mode Setting (KMS) if the kernel does not select the ideal resolution itself—for example, if the video signal is routed via a KVM switch. If you also want to specify the color depth (such as 24 bit) and the frame rate, the syntax looks as follows: `video=1280x800-24@60`.

  The setting of the graphics data only works with KMS-compatible drivers. The `video` setting usually applies to all connected monitors. If you want to change the resolution for a single monitor only, you need to specify the corresponding signal output, such as `video=VGA-1:1024x768`.
- **nomodeset**
  This option disables KMS. It prevents a notebook with an NVIDIA graphics card but without the required NVIDIA drivers from having the screen go black when the graphics system starts.
- **mitigations=auto|auto,nosmt|off**
  This option controls which protective measures the kernel should take against CPU errors (see Section 21.8).
- **dis_ucode_ld**
  This option prevents the kernel from passing microcode updates to the CPU (see Chapter 16, Section 16.9). It should only be used if there are crashes or instabilities due to such an update.

21.6.2 Symmetric Multiprocessing Options

Symmetric multiprocessing (SMP) refers to the kernel's ability to use multiple CPUs or CPU cores simultaneously. If you encounter any problems, the following options may be useful:

- **maxcpus=1**
  If you have boot problems with a multiprocessor system, you can use this option to reduce the number of processors used to 1. The value 0 corresponds to the `nosmp` option.
- **nosmp**
  This option disables the SMP functions. The kernel uses only one CPU.

- `noht`
  This option disables the hyperthreading function. Thanks to this feature, many CPUs behave as if multiple cores were available, which results in a somewhat higher computing power, although the increase is not as high as with real SMP.
- `nolapic`
  Advanced Programmable Interrupt Controller (APIC) is a procedure to forward hardware interrupts to the CPUs. In current kernel versions, APIC is always enabled. If you suspect problems with APIC, you should use `nolapic` to prevent the kernel from enabling or using the local APIC.
- `noapic`
  This option goes a little less far than `nolapic` and only disables the IO part of APIC.
- `lapic`
  This option explicitly enables APIC. This is necessary if APIC is deactivated by the BIOS, but should still be used.

21.6.3 Advanced Configuration and Power Interface Options

The *Advanced Configuration and Power Interface* (ACPI) power management system is not only responsible for powering on and off, but also for using energy sparingly, managing various hibernate modes, and more. The following sections summarize the most important options for controlling the kernel's ACPI functions:

- `acpi=on/off`
  This option (de)activates the ACPI functions in the kernel.
- `acpi=oldboot`
  This option ensures that the ACPI functions are only used during the boot process. As soon as the computer is running, however, the ACPI functions are no longer used.
- `pci=noacpi`
  This option disables the interrupt assignments made by ACPI.
- `noresume`
  This option causes existing hibernate data in the swap partition to be ignored. It's useful when the computer doesn't wake up properly, such as due to the hibernate data being defective.

21.7 Changing Kernel Parameters

Many parameters of the kernel can be changed during operation via the `/proc` file system. The following example shows how you can enable masquerading to use the computer as an internet gateway for other computers:

```
root# echo 1 > /proc/sys/net/ipv4/ip_forward
```

A more elegant way is to use the `sysctl` command, which is supplied with most current distributions. The analog command to turn off masquerading would look as follows:

sysctl

```
root# sysctl -w net.ipv4.ip_forward=0
```

`sysctl -a` returns a list of all kernel parameters along with their current settings. You can use `sysctl -p` to activate the `sysctl` settings stored in a file. Usually `/etc/sysctl.conf` is used as the filename. The syntax is described in the manual page for `sysctl.conf`. Many distributions provide for this file to be automatically analyzed and run during the init process.

21.8 Spectre, Meltdown, and Others

In the winter of 2018, it became known that CPUs from several manufacturers (Intel is particularly affected) contain serious design flaws that allow a process to read the data of other processes without permission. The first such vulnerabilities were given the illustrious names of *Spectre* and *Meltdown*. Since then, various other related problems have been discovered: *Fallout*, *Foreshadow*, *RIDL*, *ZombieLoad*, and so on (see *https://en.wikipedia.org/wiki/Transient_execution_CPU_vulnerability*).

Because it is neither technically nor economically possible to replace all affected CPUs, attempts have since been made to work around the errors in various ways. The four main starting points are as follows:

- Microcode updates for the CPU (see Chapter 16, Section 16.9)
- Changes to the compilers to generate code that prevents exploitation of the bugs
- Changes in the kernel with the same objective (besides Linux, this concerns also Windows, macOS, etc.)
- Changes to programs that are particularly affected, such as web browsers and virtualization systems

As a matter of fact, these measures have made it possible to avoid many (though not all) errors. However, the price for this is high. Benchmark tests

by Michael Larabel have shown that the performance of computers secured in this way has dropped considerably, depending on the application and CPU (see *https://www.phoronix.com/search/Spectre*).

This leads to the absurd situation that CPU manufacturers are not held responsible for their design flaws, but even indirectly profit from them: the reduced computing power makes hardware updates necessary earlier than planned.

Am I affected? To determine which problems affect your machine or CPU and which vulnerabilities are fixed by the kernel, you should take a look at the files in the `/sys/devices/system/cpu/vulnerabilities` directory:

```
user$ cd /sys/devices/system/cpu/vulnerabilities
user$ grep  *
itlb_multihit:KVM: Mitigation: Split huge pages
l1tf:              Mitigation: PTE Inversion; VMX: conditional cache
                               flushes, SMT vulnerable
mds:               Mitigation: Clear CPU buffers; SMT vulnerable
meltdown:          Mitigation: PTI
mmio_stale_data:   Mitigation: Clear CPU buffers; SMT vulnerable
spec_store_bypass: Mitigation: Speculative Store Bypass disabled via
                               prctl and seccomp
spectre_v1:        Mitigation: usercopy/swapgs barriers and __user
                               pointer sanitization
spectre_v2:        Mitigation: Retpolines, IBPB: conditional, IBRS_FW,
                               STIBP: conditional, RSB filling
srbds:             Mitigation: Microcode
tsx_async_abort:   Mitigation: Clear CPU buffers; SMT vulnerable
```

You can find many more details in the `spectre-meltdown-checker.sh` script:

```
user$ git clone https://github.com/speed47/spectre-meltdown-checker.git
user$ cd spectre-meltdown-checker
user$ sudo ./spectre-meltdown-checker.sh
Checking for vulnerabilities on current system
Kernel is Linux 5.4.0-148-generic
CPU is AMD Ryzen 5 3600 6-Core Processor

Hardware check
* Hardware support (CPU microcode) for mitigation techniques
  * Indirect Branch Restricted Speculation (IBRS)
    * SPEC_CTRL MSR is available:  YES
    * CPU indicates IBRS capability:  NO
    * CPU indicates preferring IBRS always-on:  NO
    * CPU indicates preferring IBRS over retpoline:  YES
```

```
...

CVE-2017-5753 aka 'Spectre Variant 1, bounds check bypass'
* Mitigated according to the /sys interface:  YES  (Mitigation: ...)
* Kernel has array_index_mask_nospec:  YES
* Kernel has the Red Hat/Ubuntu patch:  NO
* Kernel has mask_nospec64 (arm64):  NO
* Kernel has array_index_nospec (arm64):  NO
> STATUS:  NOT VULNERABLE
...
> SUMMARY: CVE-2017-5753:OK  CVE-2017-5715:OK  CVE-2017-5754:OK  ...
           CVE-2018-3639:OK  CVE-2018-3615:OK  CVE-2018-3620:OK  ...
           CVE-2018-12126:OK CVE-2018-12130:OK CVE-2018-12127:OK ...
           CVE-2019-11135:OK CVE-2018-12207:OK CVE-2020-0543:OK
Need more detailed information about mitigation options? Use --explain
A false sense of security is worse than no security at all, see
--disclaimer
```

Disabling protection mechanisms

By default, almost all currently available protections are active in the kernel. The only exception concerns hyperthreading or *simultaneous multithreading* (SMT). This is a method that allows a CPU to execute more threads in parallel than real cores are available. Although SMT does provide a significant performance boost, it should actually be disabled for security reasons. Currently, however, attacks aimed at it are relatively difficult to implement.

For each of the protection methods implemented in the kernel, there are separate kernel parameters to disable the respective method: `nospectre_v1`, `nospectre_v2`, `nopti`, and so on. In addition, there is the `mitigations` parameter, which provides three settings:

- **`mitigations=auto`**
  In the default setting, all available protection measures are activated. SMT remains active, however.
- **`mitigations=auto,nosmt`**
  This setting disables SMT and is even safer. However, if your CPU supports multithreading, the performance of applications with many threads drops significantly again.
- **`mitigations=off`**
  This setting deactivates all protective measures and makes your computer faster. However, this is only recommended if you feel secure against possible attacks. In some cases, this can certainly be advocated. Although Spectre, Meltdown, and other such vulnerabilities have been

theoretically proven to date, there is currently hardly any known malware that actually exploits these vulnerabilities.

In this respect, it's tempting to do without protection in certain use cases, such as on development or laboratory computers that frequently run CPU-intensive applications. However, this option is completely taboo on all servers running virtual machines of different owners or customers.

To try out the `mitigations` settings, you can open the editor for the relevant GRUB menu item during the boot process with E and add `mitigations=...` at the end of the `linux` line.

To set the option permanently, you must change the `GRUB_CMDLINE_LINUX_DEFAULT` line in `/etc/default/grub` and then update the GRUB configuration using `update-grub` or `grub2-mkconfig`.

PART VI

Server Configuration

Chapter 22
Server Installation

Several chapters on the server application of Linux start at this point. Most of these chapters provide enough material to fill an entire book each—and indeed, there are various books that deal only with Apache, MySQL/MariaDB, Postfix, or Samba.

In this respect, this section of the book should be regarded as an introduction rather than a complete reference. Nevertheless, about 250 pages is quite sufficient to give you a first idea of how to set up a web and mail server or a Windows-compatible file server on Linux.

This chapter starts with a basics section that summarizes server-specific features to be considered during the installation: these features range from the correct hostname setting to BIOS- or EFI-ready RAID-1 configuration. This is followed by specific instructions for installing RHEL, Oracle Linux or other clones, and Ubuntu Server.

Finally, this chapter provides tips on using Linux in the cloud. As a matter of fact, more and more Linux servers are no longer running directly on real hardware, but are using cloud infrastructure. (Of course, every Linux installation ultimately runs on "real" hardware. Increasingly abstract interfaces, however, make this hardware configurable at the click of a mouse.)

Cloud offerings give you as the administrator much greater flexibility to respond to peak loads during ongoing operations or to increase the size of your disks as needed. The often-promised cost savings, on the other hand, are very doubtful, especially as the calculation of cloud offers is becoming more and more confusing every year. The price-performance ratio of a traditional root server is often much better, but there, of course, the versatility and convenience of cloud features are missing.

22.1 Basic Principles

Basically, if you have come this far and have read at least a few chapters, you are probably already fairly familiar with Linux. How does an "ordinary" installation differ from a server installation?

Text only

In general, you can of course also install server packages on any desktop installation and thus turn any desktop Linux into a server. But this is an unusual approach. Instead, most people try to keep server installations as

lean as possible for two reasons: to achieve maximum efficiency and to improve security. (The motto here is that what is not installed cannot cause any security problems.)

For these reasons, most server installations run without a graphical user interface, so they are operated exclusively in text mode. This is only a limitation at first glance. In fact, you will usually administrate your server via SSH or possibly use a web interface like Cockpit. A desktop system like GNOME, which takes up space on the disk and in the memory, is superfluous.

22.1.1 Installation Method

The installation of a Linux server depends heavily on the type of hardware on which the server is supposed to run. The following list contains the most important variants:

- **A "real" computer at home or at a company**
  In a way, this is the simplest case. Operation is via a directly connected keyboard and monitor and is even possible if there are network problems. You can replace defective hardware yourself, install additional hard disks or SSDs yourself. For installation, you can use the installation programs of the respective distribution.
- **A root server with a server provider**
  In small companies (and sometimes in large ones as well), web or mail servers run rarely on the company premises, but instead at a server provider. This primarily has the advantage that the network connection is much better. But the separation from the local network also comes with security benefits. (The term *root server* originates from the fact that you can log in directly or indirectly as `root`; that is, you have unrestricted admin rights on the computer.)

  As for the installation, there are two variants: With some providers you can perform an installation as on a local computer, but the setup program is operated via VNC (i.e., through a network connection). Other providers offer you a choice of preconfigured minimal installations for some distributions, which you can then use as a starting point for your further configuration work.
- **Virtual machine**
  A server installation on a virtual machine is mostly done in the same way as a desktop installation. You can control the installation from a window of the virtualization system. With many virtualization systems, this also works via a network connection. In this case, you control a virtual machine in a local program that runs on a computer (virtualization

host) that may be thousands of miles away from your office. Behind the scenes, VNC or a similar protocol is often used in this case as well (e.g., Spice).

- **Cloud instance**
  You can also run all server-ready Linux distributions in the cloud. In this case, you will get access to a preconfigured minimal installation, setting in advance the CPU power, RAM size, and virtual disk size as per your requirements.

  RAID and LVM are unnecessary because these functions are already available at the cloud level. A division into multiple file systems (`/`, `/home`, `/var`, etc.) is also often omitted. Even the firewall is frequently eliminated in favor of cloud-specific security solutions. In this respect, a pleasant minimalism prevails in the cloud configuration.

  But the hurdles are elsewhere: before you can put a cloud instance into operation, you have to deal with the often confusing (virtual) hardware variants and billing options of the respective provider. Each cloud provider pursues its own strategies. Sometimes, unfortunately, one gets the impression that the primary goal is to confuse the customer as much as possible.

Web Hosting and Managed Server

If you're interested in creating a web presence for a company (possibly with your own mail addresses), then you will find countless ready-made web hosting offers for this purpose. Depending on the provider, these are also referred to as *managed servers*.

Specifically, this means that you don't have to worry about installing and configuring Linux. Rather, you can only influence the server via a web interface as well as by uploading files. You do *not* have admin rights, so it is just not a *root server*. Usually, you do not even know which Linux distribution is used behind the scenes.

These kinds of web hosting offerings are absolutely justified but are not a topic for this book. I assume that you want to administrate your server yourself, which requires a higher level of expertise; in return, you get more flexibility.

22.1.2 Host Name

For Linux installations in the private sector, the host name is largely irrelevant. Even on local networks, setting the hostname is rarely a major challenge: you simply choose a name that is not yet in use. Depending on the

network, you may also need to specify the name of a local domain. This, in turn, results in a complete hostname, such as `mynewhostname.localnet`.

Much more care is required when you set the host name of a server that is to be accessible on the internet as a web or mail server. Usually you have reserved a domain for your company or for your organization, such as `example.com`. Although it is possible to use this domain name directly as a host name, this is not recommended! Instead, you must give your server a specific host name *within* that domain. If you can't think of anything better, then choose `host1.example.com`, for example. Of course, this also requires you to add a corresponding A record in the DNS configuration form. This entry (the *A resource record*) specifies which IPv4 address `host1.example.com` is connected to. For IPv6, you also need an AAAA entry.

Especially users who set up the first server for a small web presence often overlook the fact that countless servers can be connected to one domain name. (Think of big companies like Google or IBM.) As long as you only run one server (and do not configure it as a mail server), a domain name like `example.com` will work as a host name if need be. But by the time you configure the second server for the same domain, you will realize that you have reached a dead end.

DNS Configuration for Mail Servers

I discuss the required DNS settings for mail servers in detail in Chapter 26, Section 26.1. There you will find a specific example with all DNS settings for a web and mail server with IPv4 and IPv6. Mail servers also have some special features, including the so-called reverse DNS entry.

22.1.3 RAID/LVM Setup

You should already be familiar with the basic principles of RAID and LVM. (If not, please take another look at Chapter 18, Section 18.15 and Section 18.16.) The question as to whether it makes sense to use RAID and LVM on your server depends largely on the system requirements—that is, whether you are using real hardware or not:

- Whether it is a machine you maintain yourself or a root server with a provider, I strongly recommend using RAID-1 and LVM. Reputable server providers will provide you with at least two hard disks or SSDs for such a configuration as part of their standard package. (If large amounts of data and correspondingly more hard disks/SSDs are involved, RAID-5 or RAID-6 is a good alternative to RAID-1. However, I will focus on RAID-1 here.)

- On the other hand, if your server is running in a virtual machine or in the cloud, you can do without RAID. Redundancy is unnecessary because you can now (hopefully) assume that the disks of the virtualization host or the cloud system are themselves protected by RAID.

  The question of LVM is not quite as clear-cut. Typically, virtualization and cloud platforms offer the ability to easily scale up virtual disks. The use of the additional storage within the virtual machine or cloud instance is slightly more complicated than with LVM and requires a reboot; whether this argument is sufficient to justify the use of LVM is something you have to decide for yourself. (Many cloud providers take the decision away from you and offer no options at all.)

Regardless of whether with or without RAID or LVM, the golden KISS rule also applies to the server configuration, which means: *Keep it simple, stupid!* So it's all about finding the simplest possible setup that just fits your own desires. KISS

In my daily work, I surprisingly often have to deal with servers or virtual machines that use a BIOS for the boot process (i.e., no EFI; see also the following note box). In such cases, I use a setup with a boot partition in a RAID-1 array and an LVM system in a second RAID-1 array (see Figure 22.1). In the LVM area, I then set up logical volumes as needed for swap space, the root file system, and other file systems as needed (`/home`, `/var`, etc.).

Theoretically, GRUB should also be able to cope with a boot partition within the LVM system; however, the installation programs of many distributions refuse such a configuration.

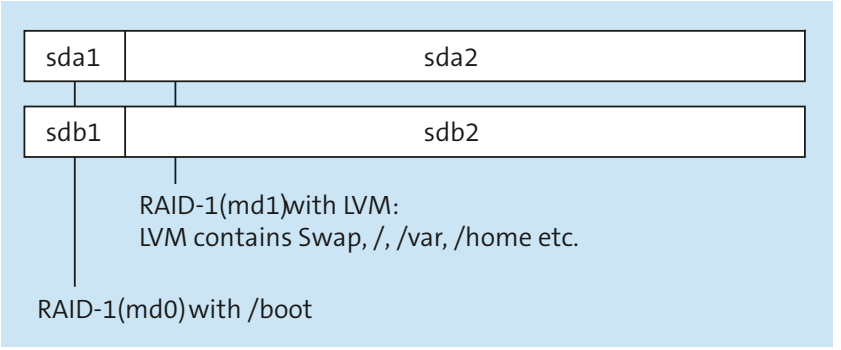


Figure 22.1 Simple Server Configuration with RAID-1 and LVM for Server with BIOS

If, on the other hand, the server supports EFI, an EFI partition is also required (see Figure 22.2). Unfortunately, unlike the boot partition, the EFI partition must not be in a RAID-1 array. For reasons of fail-safety, it's useful

to set up an EFI partition on both hard disks or SSDs and mirror its contents using a Cron job.

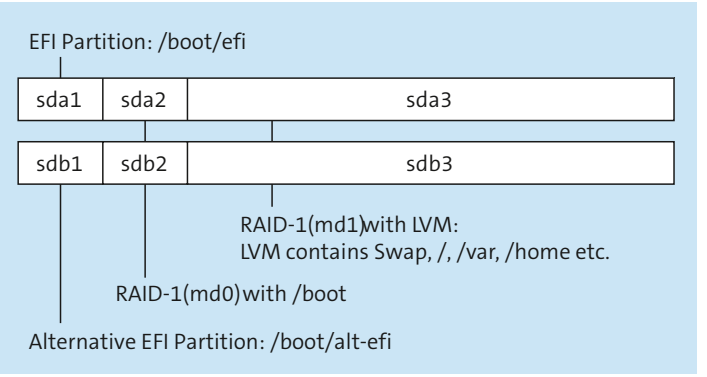


Figure 22.2 Simple Server Configuration with RAID-1 and LVM for Server with EFI

EFI Test

EFI has been the standard on PCs and notebook computers for approximately a decade. Server hardware is somewhat more conservative in this respect. Surprisingly often, an old-fashioned BIOS is still used there. This is also true in virtual machines: if there is any EFI support at all, you have to find the well-hidden option first, install add-on packages, and so on.

Before you start complaining about the old-fashioned BIOS, here's just a quick note: The benefit of EFI on a server is small. However, EFI makes it much more difficult to configure the server in a fail-safe manner (discussed ahead). So the good old BIOS can definitely save you some work. To easily determine whether EFI is available or not on Linux, you should check if the `/sys/firmware/efi` directory exists. If this directory is missing, the computer was started with BIOS.

22.1.4 Improving Reliability

The RAID setup protects you from data loss when a hard disk or SSD fails. However, there is no guarantee that your server will boot after replacing a defective disk. If you don't put any effort into the configuration, the boot behavior is a lottery game:

- If you are lucky, the second disk will be defective. Once you have replaced the hard disk/SSD, the computer finds all the necessary boot files on the first hard disk when restarting. The computer reboots with a degraded RAID system. You can log in via SSH, configure the replaced hard disk, and restore the RAID array (see Chapter 18, Section 18.15).

- Unfortunately, sometimes the first disk fails. After it has been replaced, the server looks for a boot sector or EFI partition on the first hard disk/SSD, finds nothing there, and hangs. For further repair work, you then need direct access to the computer or the assistance of your server provider. (Many providers offer an option to boot the server via a network image for such cases. You then perform the repair work in this emergency image. But this is more complicated, and your server will be offline during the repair work.)

It is therefore preferable to have a configuration that ensures that the server boots again without further ado after a hard disk or SSD swap. In many cases, this requires a little manual work after the actual installation.

BIOS

If the boot partition of a BIOS server is mirrored across both hard disks/SSDs using RAID-1, only a tiny detail determines whether the server can also be booted from the second disk: GRUB must be installed in the MBR of both hard disks/SSDs. The Debian and Ubuntu installers usually take care of this detail. In other distributions, a tiny command is sufficient to improve the poor configuration. You must replace `/dev/sdb` with the device name of the second disk:

```
root# grub2-install /dev/sdb
```

If it is indeed necessary to replace a hard disk or SSD during the server's lifetime, do not forget to repeat this command after repairing the RAID system. To ensure that a second defect is as smooth as possible, GRUB must again be written to the boot sector of the new hard disk/SSD.

EFI

Things get a bit more complicated once EFI is in play. The EFI partition must not be located in a RAID array. For this reason, you must provide *two* EFI system partitions (ESPs) during the installation: one on each hard disk/SSD (see Figure 22.2). One of them is mounted in the file system as `/boot/efi` as usual.

For the second ESP (`/dev/sdb1` in the following example), some manual work is required after installation. You need to set up the `/boot/alt-efi` directory and determine the file system ID:

```
root# mkdir /boot/alt-efi
root# mount /dev/sdb1 /boot/alt-efi
root# blkid /dev/sdb1
/dev/sdb1: UUID="BD7C-49AF" TYPE="vfat" PARTUUID="8088034d-3952-..."
```

Then you must modify `/etc/fstab` as in the following pattern. The `nofail` option for the `/boot/efi` and `/boot/alt-efi` mount directories is crucial here. If a disk actually fails, only one of the two file systems can be activated.

The boot process should therefore not get stuck if the second file system is missing:

```
# File /etc/fstab
...
# entries for /boot/efi and /boot/alt-efi
UUID=BD7B-4B73  /boot/efi      vfat  nofail,umask=0077  0  1
UUID=BD7C-49AF  /boot/alt-efi  vfat  nofail,umask=0077  0  1
```

You can use the following command to copy the contents of `/boot/efi` to the second EFI partition. In the future, changes are expected only very rarely in updates. To synchronize the contents of the two directories in the future, you must simply set up a script in `/etc/cron.daily` that also contains the `rsync` command:

```
root# rsync -a /boot/efi /boot/alt-efi
```

Now you still need to inform the EFI system that it must take the new EFI partition into account during the boot process if necessary. The required command looks as follows:

```
root# efibootmgr  --create --disk /dev/sdb --part 1 \
                  --label 'ESP on sdb' --loader '\EFI\ubuntu\
shimx64.efi'
```

Test the Configuration

Needless to say, it's a good idea to test this rather complex setup by simulating an emergency. In a virtual machine, you simply remove the first disk. (For the boot process, the first hard disk/SSD is more critical than the second one.)

On a real computer, you'll want to disconnect the cable to a hard disk or SSD. After that, you need to check if your server still starts. Finally, try to restore the now-degraded RAID system with a new virtual or real disk. (For detailed instructions, see Chapter 18, Section 18.15.)

If your server is hosted by an external service provider, a test is unfortunately impossible. In my work experience, I have only ever had to deal with a defective hard disk that failed after around three and a half years (root server with BIOS). The hard disk can be replaced and the system restored with a downtime of just a few minutes. Although it took hours afterward until the RAID system was redundant again, the server was running normally during this time—albeit a little slower than usual.

22.2 Red Hat Enterprise Linux

RHEL

RHEL (*https://www.redhat.com*) is the most commercially successful Linux distribution. It is used, for example, in banks, in insurance companies, and on mainframes—in other words, whenever stability and professional support are top priorities. In addition to the core RHEL product, Red Hat offers various RHEL extensions and specialty products, such as Red Hat Storage Server, JBoss Middleware (Java EE), and the Red Hat Virtualization System.

RHEL is basically based on a Fedora version that was released about six months to a year earlier. With RHEL 9, this was Fedora 34. However, there are of course fundamental differences between Fedora and RHEL: the Enterprise version does not include any functions that are not yet fully developed, because stability has the highest priority in the professional environment. For this reason, in RHEL you have to make do with older versions of various programs. On the other hand, the support period is much longer and amounts to between 7 and 13 years. Finally, Red Hat Subscription Management (RHSM) helps with the centralized maintenance of multiple RHEL installations.

Unlike the other distributions presented in this book, RHEL is freely available only in exceptional cases. Installation media and updates are only available to paying customers or users of a free license.

Version numbers

RHEL uses a versioning scheme that differs from the schemes of other distributions: there is a "major" new RHEL version (major release) approximately every three to five years. Currently, RHEL 9 is the latest version. This version was presented in May 2022. RHEL 7 and RHEL 8 (introduced in April 2014 and May 2019 respectively) are still being maintained.

Within each major RHEL release, there is a new minor release (version 9.1, 9.2, etc.) once or twice a year. In the course of such updates, there are also new installation media that are compatible with current hardware. Fundamental new functions are only implemented in exceptional cases, and if they are, then usually under the designation *technical preview* (in plain language: without official support).

The special feature of the minor releases is that they do *not* require a new installation. So, for example, if you install RHEL 9.2 and then regularly perform all updates, you will gradually get RHEL 9.3, RHEL 9.4, and so on. A new installation is only required to upgrade to the next major release.

However, you shouldn't expect any miracles from this update service. For many components, only bug fixes are performed, but no version updates. One of the few exceptions is Firefox, which is updated to the latest ESR

version approximately every nine months. Other components like GNOME and LibreOffice also make version jumps from time to time.

A huge improvement compared to earlier versions is the AppStream concept that was introduced in version 8. For important programming languages and server services, you can choose between modules with different version numbers (see Chapter 16, Section 16.3). So, for example, you don't have to stick with PHP version 7.2, which was originally shipped with RHEL 8, but can alternatively install version 7.3, 7.4, or 8.0. In the past, such version updates were only possible using external package sources.

AppStream combines the best approaches from both worlds. Administrators don't have to worry about sudden updates that challenge system stability or compatibility, but still have the option to upgrade to modern software versions.

Update period

Red Hat guarantees a maintenance period of ten years from the date of the major release. For an additional charge, there is a two-year phase after that in which critical security issues are fixed. The following web page provides a detailed breakdown of the various phases in the lifecycle of a RHEL release: *https://access.redhat.com/support/policy/updates/errata*.

Free RHEL Licenses

In the past, software developers have been able to use RHEL free of charge (but not in production) as part of the Red Hat Developer Program. In December 2020, Red Hat liberalized access to free licenses. For each developer license, 16 installations (whether on physical hardware or as virtual machines) can be used freely —even in productive operation:

- *https://developers.redhat.com/products/rhel/download*
- *https://www.redhat.com/en/blog/new-year-new-red-hat-enterprise-linux-programs-easier-ways-access-rhel*

Basically, this offer allows smaller companies to use RHEL for free. In actual use, however, this is hardly practicable. The use is limited to one year. It is possible to create a new account afterward and connect the installations to the new account, but the process is extremely cumbersome.

For paying customers, Red Hat's benefits are indisputable. On the other hand, if you want to use RHEL-compatible systems for free, whether as a developer, for teaching purposes, or in production, the RHEL clones described in the following section are definitely more attractive. You won't get any commercial support without money, but with the clones you at least don't have to bother with the confusing administration of licenses or subscriptions.

Red Hat and IBM

Red Hat was acquired by IBM in 2018. However, as Red Hat continues to operate fairly independently within the IBM group, I write in this book that *Red Hat* announces this decision or introduces that release, not IBM. It should be clear that IBM is ultimately the owner and pulls the strings.

22.2.1 CentOS

Of course, Red Hat must also comply with the rules of the GPL and make the source code of its products available. That is why Red Hat clones such as AlmaLinux, CentOS, Oracle Linux, or Rocky Linux are legally permissible. All clones presented ahead are based on the original source code from Red Hat!

However, the code cannot be transferred unchanged: *Red Hat* is a protected trademark. Any packages that contain Red Hat–specific strings, logos, or images must be modified. Some Red Hat–specific packages, such as those for accessing Red Hat Subscription Management, are removed entirely. The modified packages must be compiled, tested, and finally bundled in the form of new installation media. To support EFI Secure Boot, a Microsoft-managed key is required.

All this costs time and effort and explains why it often takes weeks after the release of a new RHEL version for the same version to be offered by the various clones. (CentOS was particularly slow in this regard.)

CentOS

CentOS (*https://www.centos.org*) was *the* RHEL clone par excellence until 2020. CentOS was binary-compatible with RHEL, but unlike RHEL, it was available free of charge, including all updates. Of course, you had to cut back on commercial support: there was none. Still, CentOS was a great way for administrators with Red Hat or Fedora experience to use Red Hat compatible servers on projects with small budgets—and this option was used millions of times!

The relatively small CentOS team was somewhat surprisingly acquired by Red Hat in January 2014. Initially, this cooperation worked well until Red Hat unexpectedly announced the end of CentOS in its previous form in 2020. The discontinuation of CentOS 8 hit Linux administrators out of the blue. CentOS 8 had only been around for 15 months at that time. Many administrators had just started migrating initial production environments to CentOS 8, only to realize that they had suddenly hit a dead end.

CentOS Stream

Red Hat promotes the new *CentOS Stream* product as the successor to *CentOS*. Although CentOS Stream is a free RHEL clone just like CentOS, that's where the similarities end:

- First, updates planned for RHEL will be released for CentOS Stream first. CentOS Stream is thus a testing environment for Red Hat. If a problem does occur during an update, CentOS Stream users will discover this before paying RHEL customers are affected.
- Second, CentOS Stream has a much shorter lifetime than RHEL. While CentOS promised 10 years of updates in the past like RHEL, these will only be available for CentOS Stream for about three to four years. (This period is not set in stone, though. It depends on how quickly Red Hat completes each next RHEL release.)

CentOS Stream is well-suited for software developers as well as for teaching, but it is largely unsuitable for use on a production server.

22.2.2 AlmaLinux, Oracle Linux, and Rocky Linux

In 2020, many long-time CentOS users were suddenly faced with the question, "What now?" Red Hat would of course have liked to gain a lot of new, paying customers—but that is not to be expected: companies that have the necessary budget use RHEL or another commercial Linux distribution anyway.

Switching to Ubuntu or Debian is also not very attractive for many CentOS fans: on the one hand, this requires learning a lot of new details, and on the other hand, the lifecycle of these distributions is significantly shorter. *The* argument for RHEL, CentOS, and the like was always the long-term updates.

In this section, I therefore present three potential CentOS successors, sorted by their age. While AlmaLinux and Rocky Linux were just two years old in the summer of 2023, Oracle Linux was introduced as a commercial RHEL clone more than 17 years ago.

Oracle Linux

Oracle has been active in the RHEL clone market since 2006, but with a completely different concept than CentOS. On the one hand, Oracle has marketed its Linux primarily as a commercial offering; on the other hand, Oracle is trying to differentiate itself from RHEL with additional features. These include a newer kernel (in marketing jargon, the Unbreakable Enterprise Kernel), Ksplice kernel updates during operation, and better support for the `btrfs` file system codeveloped by Oracle.

Initially, Oracle Linux, like RHEL, was only available to paying customers, albeit at much lower prices than RHEL. Since 2012, Oracle Linux including all updates can be obtained free of charge and is thus on par with AlmaLinux and Rocky Linux.

In Oracle's favor is the fact that the update supply has worked excellently (and much better than with CentOS!) over the last 15 years. One argument against this is that Oracle has not had a happy relationship with open-source projects in the past. Java, MySQL, and OpenOffice top the list of projects acquired by Oracle, which promptly caused conflicts and forks.

AlmaLinux

Only a few days after Red Hat's announcement to stop CentOS in its current form, the CloudLinux company announced that it would build a RHEL-compatible distribution and maintain it in the long term. The first version was introduced at the end of March 2021, and further responsibility was placed in the hands of the newly created AlmaLinux OS Foundation. CloudLinux promises to provide $1 million per year for the project.

In the first two and a half years, AlmaLinux made an extremely professional impression, both as a distribution and as an organization. No other RHEL clone was able to deliver updates as quickly as AlmaLinux. Among the clones presented here, AlmaLinux is my personal favorite.

Rocky Linux

At about the same time as CloudLinux, former CentOS cofounder Greg Kurtzer announced another RHEL clone called *Rocky Linux*. It took a little longer than AlmaLinux to release the first version. Since then, however, Rocky Linux has also provided its users with regular updates. Rocky Linux enjoys the support of Amazon, Google, and VMware, among others.

The agony of choice

At this point, it's impossible to make a recommendation for any of the three distributions presented here. All three vendors promise updates over the same period as RHEL (i.e., 10 years), and all three promise binary compatibility at the application level. (With Oracle, you have the choice between a kernel maintained by Oracle or the original kernel based on the RHEL code. By default, the Oracle kernel is used.) Oracle Linux is supported by the fact that this distribution has been around since 2006. During this period, Oracle's maintenance of the distribution has been excellent. In this regard, it's a huge advantage that Oracle treats Linux paying customers and free users equally when it comes to providing updates.

Personally, I have had consistently positive experiences with Oracle Linux over the past year. But of course, no one can stop Oracle from focusing on the commercial variant of Oracle Linux in the future and ending free access to updates.

AlmaLinux and Rocky Linux leave an excellent impression from today's perspective, but maintaining an enterprise distribution requires staying power. Red Hat has squandered the trust that CentOS has earned over many years virtually overnight. In contrast, it will take years for AlmaLinux and Rocky Linux to gain this trust.

For more information, visit the following pages:

- *https://oracle.com/linux*
- *https://almalinux.org*
- *https://rockylinux.org*

In general, almost all text passages that refer to RHEL in this book also apply in the same way to Oracle Linux, AlmaLinux, and Rocky Linux. The only exception is the Subscription Manager, which you only need with the original to be allowed to use the package sources.

22.2.3 Distribution Change on the Fly

All distributions provide scripts to switch from one RHEL-compatible distribution to another. These scripts change the Yum package sources and then replace all installed packages with the version of the respective distribution. After a reboot, AlmaLinux or Oracle Linux, for example, runs instead of the originally installed CentOS. For more information, visit the following pages:

- *https://github.com/oracle/centos2ol*
- *https://github.com/AlmaLinux/almalinux-deploy*
- *https://github.com/rocky-linux/rocky-tools/tree/main/migrate2rocky*

Warning

I tried two such scripts and was surprised by how well and smoothly the changeover worked. But these tests by no means prove that the distribution change will go through without a hitch in every case. Of course, you must make a full backup beforehand, and of course you must be able to perform a fresh installation based on the backup if necessary!

Ideally, you should perform the distribution change as early as possible—that is, before you install countless additional packages and set up server services. This approach is useful if your cloud or hosting provider does not provide the clone of your choice. In that case, use a minimal installation as a starting point, perform a change immediately, and only then continue to set up the system.

22.2.4 RHEL versus Clones

For nearly two decades, the commercial RHEL original and its clones coexisted relatively peacefully. That changed abruptly in the summer of 2023 when Red Hat announced that it would discontinue a Git repository that contained the source code of all current RHEL packages. From now on, only paying customers will have access to the source code, which, according to Red Hat, will comply with the rules of the GPL. Red Hat or its owner IBM did not justify this step at first, but then admitted quite bluntly that the clones would disrupt their business. Besides, the clones would not contribute anything to the open-source idea (see *https://www.redhat.com/en/blog/redhats-commitment-open-source-response-gitcentosorg-changes*).

Needless to say, the open-source community sees things completely differently. I too believe that Red Hat or IBM is making a big mistake that may have a lasting impact on the RHEL cosmos. Just as users of the clones benefit from a stable, free server Linux, Red Hat benefits from having countless developers working on projects that feed directly or indirectly (e.g., via the EPEL package source) into RHEL (see *https://www.jeffgeerling.com/blog/2023/dear-red-hat-are-you-dumb*).

The future of clones

When I wrote these lines in August 2023, it was completely unclear what impact Red Hat's decision would have on the future of clones. Oracle and Rocky Linux hope that thanks to the GPL there will still be ways to get the source code of the packages. SUSE is considering creating another RHEL fork.

AlmaLinux has decided to abandon one-to-one compatibility and process patches for RHEL packages from public sources itself. AlmaLinux would then "only" be compatible at the program level (*application binary interface*), which is probably perfectly sufficient for most applications (see *https://almalinux.org/blog/future-of-almalinux*).

Personally, I believe that there is a great need for RHEL clones and that they will continue to exist in one way or another, even in the face of (presumably increasing) resistance. But Red Hat, of course, has done a wonderful job of offering fear, uncertainty, and doubt. The trust with which many administrators previously used RHEL clones is fading. Moreover, there is no telling what else Red Hat plans to do to hinder the clones.

Of course, this shot can also backfire if administrators increasingly turn to Debian and Ubuntu. (I've been running Ubuntu LTS servers for years and have been extremely happy with them.)

Is a clone good enough?

Aside from the current dispute, it must be clear to you that a RHEL clone is not and never was a full replacement for RHEL:

- **Support**
  You do not receive any commercial support. When problems arise, you need to rely on forums, mailing lists, and self-help. There are optional support offers for most clones, which are not free, but cheaper than Red Hat licenses. With Oracle, it has always been possible to switch to the commercial branch of the distribution.
- **CPU architectures**
  While some RHEL versions are also available for fancy architectures and hardware platforms (IA-64, Power7 and Power8, and IBM zSeries), most clones only support the x86_64 architecture and possibly the ARM platform.
- **Updates**
  It usually takes a few days for security updates released by Red Hat to be available to clones. For the update supply, you are dependent on the (often relatively small) organizations or companies that take care of it. There is no guarantee that you will really receive updates for 10 years as promised. (Of course, it's also conceivable that Red Hat could go out of business and no longer deliver any updates. But that's many times less likely than for most clone vendors, perhaps aside from Oracle.)

From my professional experience, clones are good enough for many tasks. This is especially true when budgets are severely constrained—whether in research, nonprofit organizations, or small businesses. On the other hand, if you have the IT budget of a large enterprise at your disposal and the data processed on the servers is of great economic importance, you will likely end up with Red Hat.

22.2.5 Installation

Installation program

RHEL and its clones use the same installer as Fedora (see Figure 22.3). Its operation is described in more detail in Chapter 3, Section 3.2.

In contrast to Fedora, you need to consider the following peculiarities:

- The partitioning tool suggests the `xfs` file system for formatting all partitions or logical volumes. If you prefer a different file system, you must explicitly set the appropriate options. The little-thought-out user guidance of the partitioning program doesn't exactly make this easy.
- In the **Software Selection** item, you can define the scope of the installation. By default, the **Server with GUI** option is selected. This option installs the GNOME desktop, various base packages, and the SSH server. If you do not need a graphical user interface, you can choose the **Minimal Installation** variant. The installation size is then only about 2 GiB.

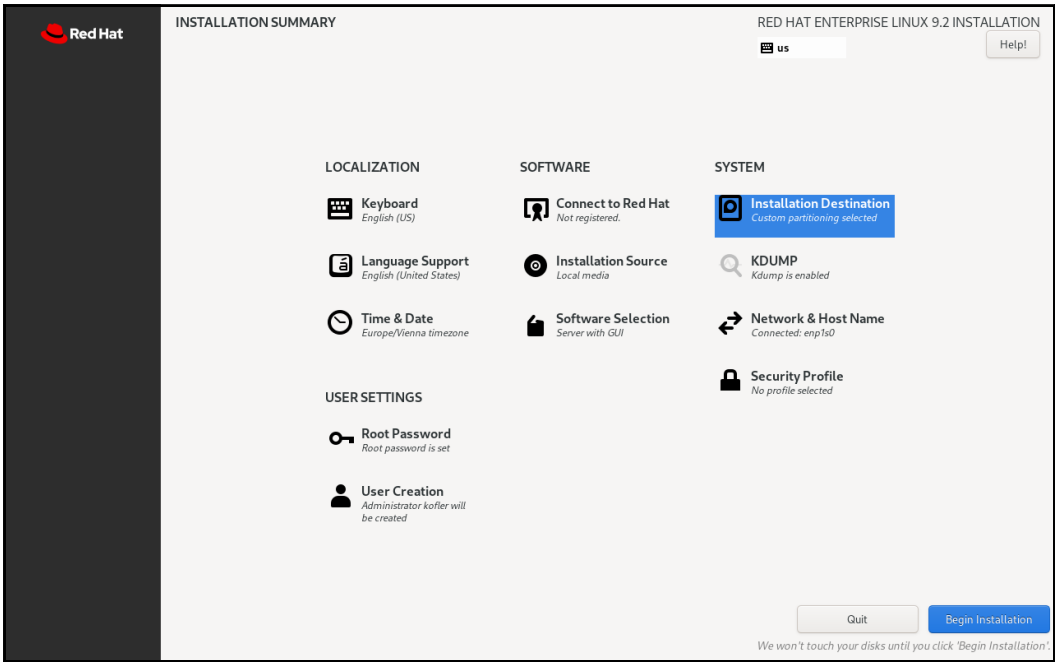


Figure 22.3 RHEL: Same Installer as Fedora

RAID and LVM

To achieve a setup like that shown in Figure 22.1, you can run the **Installation Destination** command in the main screen, click both hard disks/SSDs in the subsequent dialog (a checkmark must be displayed for both disks), and then select the **Customized** configuration variant. The misleadingly labeled **Done** button takes you to the partition editor. There, you should proceed as follows:

- **Boot**
  Use the plus button to add a new file system, select `/boot` as the mounting point, and set a suitable capacity (e.g., 1 GiB). In the details dialog, you then select the **RAID** device type and set the **RAID1** RAID level.
- **Swap**
  Use the plus button to add the next entry, but this time set **Swap** as the mounting point and the size you want. In the details dialog, you must set the device type to **LVM**. The **Change** button enables you to set RAID level 1 for the volume group with the default name `rhel`.
- **Root**
  Select mounting point / for the third entry. If you leave the **Capacity** field empty, the installer uses all the space that's still available. **LVM** should already be preset in the details dialog (see Figure 22.4). You can use the

Modify button to check the RAID level again. The setting selected during swap configuration should automatically apply to the root file system as well.

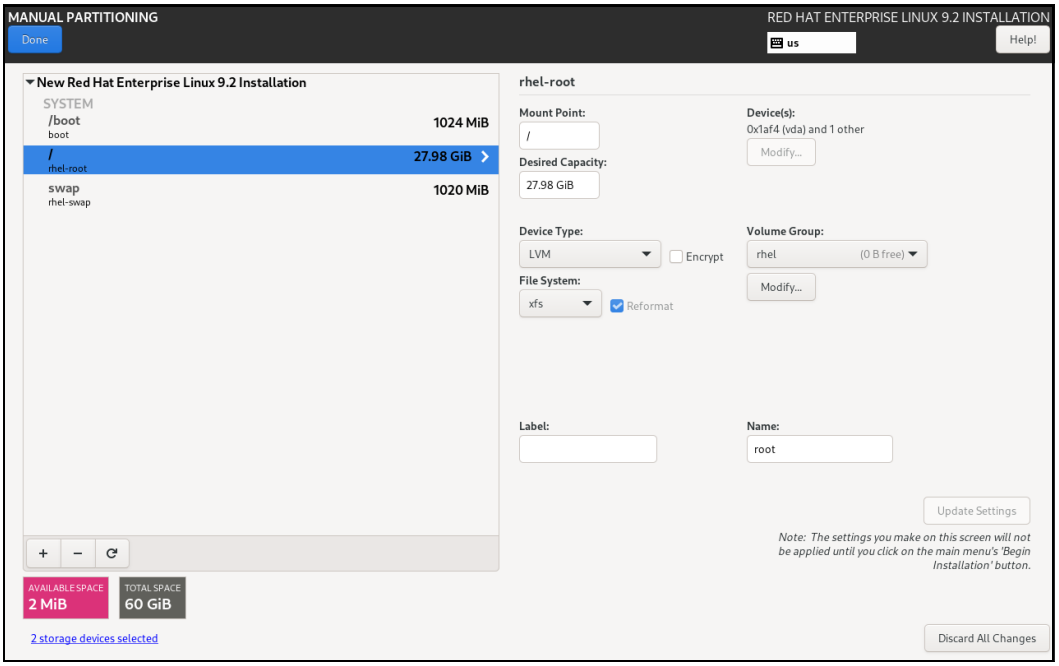


Figure 22.4 LVM and RAID Partitioning for Server with BIOS

If the server uses EFI instead of BIOS, the procedure for the setup from Figure 22.2 looks a little different:

- **EFI partition**
  Use the plus button to add a new file system. Use `/boot/efi` as the mounting point; a size of 1 GiB is sufficient. The installer is smart enough to provide a normal partition for the file system without RAID or LVM.
- **Alternative EFI partition**
  This time, instead of `/boot/efi`, you need to specify the `/boot/alt-efi` path. Select **Standard Partition** as the device type and **EFI System Partition** as the file system. Use the **Modify** button to specify that only the second hard disk is allowed as the location for this partition.
- **Boot, Swap, and Root**
  You can set up the other entries as for a BIOS server.

Root password, user account

Once partitioning is complete, the actual installation begins, during which you can set the root password and set up a user account.

RHEL subscription

In the **Connection to Red Hat** item, enter your Red Hat account login information to register the installation. However, you can also perform this step later in the Subscription Manager. This point is omitted for RHEL clones.

Kickstart installation

Instead of performing the installation manually, you can also automate this process. This is convenient if you need to perform dozens of similar installations. A detailed description of this procedure can be found at the following page: *https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/9/html/performing_an_advanced_rhel_9_installation/index*.

22.2.6 Registering the RHEL Installation

If you have installed a RHEL clone, you are now done and can immediately perform a first update using `dnf update`. However, with the original RHEL, the `dnf` command is locked until you register your system. If you have a graphical user interface on RHEL, you can start the Red Hat Subscription Manager in it (see Figure 22.5).

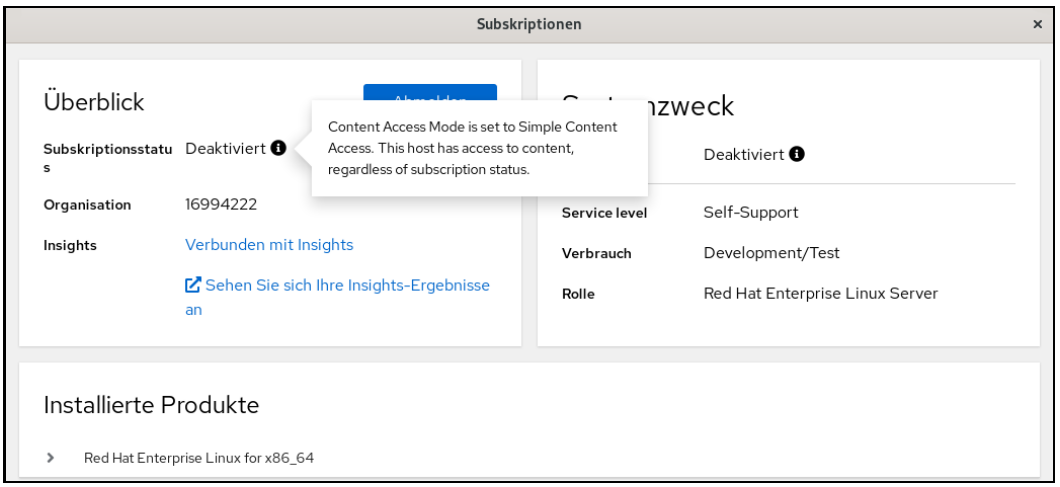


Figure 22.5 Red Hat Subscription Manager

If Subscription Manager shows the **Disabled** status, this is not necessarily a cause for concern. Red Hat introduced the new Simple Content Access mode in 2022 to simplify license management. (That would have been a good idea, but unfortunately, the current implementation is just as confusing as the old model. In general, Red Hat makes dealing with subscriptions and licenses about as accessible as Kafka's castle.) According to this new mode, `dnf` works as soon as a connection to an active Red Hat account can be established (see *https://access.redhat.com/articles/simple-content-access*).

If you've performed a minimal installation or Subscription Manager is complaining, you can also perform the registration in text mode. Moreover, you can enter the same login data as on the *https://access.redhat.com/management* website for the username and password:

```
root# subscription-manager register --username <yourname> --password <pw>
Registering to: subscription.rhsm.redhat.com:443/subscription
The system has been registered with ID: 23ce-xxx
The registered system name is: myhostname
root# subscription-manager attach
Ignoring the request to auto-attach. Attaching subscriptions is disabled
for organization xxx because Simple Content Access (SCA) is enabled.
root# dnf update
Updating Subscription Management repositories.
....
```

22.3 Ubuntu Server

Ubuntu Server is a variant of Ubuntu optimized for server operation. Especially the LTS versions of Ubuntu Server are among the most popular Linux server systems in the noncommercial sector, alongside the RHEL clones and Debian, because of the support period of five years.

Behind the scenes, Ubuntu Server is actually no different from ordinary Ubuntu: it uses the same packages and package sources. The special feature of Ubuntu Server is that there is a separate, text-based installation program especially for server use.

The installer sets up a basic system for server use without a graphical user interface. In general, it is of course also possible to use an ordinary Ubuntu installation for server use and simply install the corresponding server packages afterwards.

Ubuntu Pro

However, Canonical has also established Ubuntu Server in the commercial segment. Ubuntu Pro (*https://ubuntu.com/pro*) allows you to extend the update period to 10 years and to administrate and monitor various server installations with the optional Landscape tool (*https://ubuntu.com/landscape*).

Instead of signing a contract, you can also use Ubuntu Pro via various cloud providers. You pay for the use of Ubuntu Pro as part of your cloud billing. At AWS, an instance with Ubuntu Pro including licensing costs about 20% more than an instance with Ubuntu LTS. (The price difference varies depending on the instance.)

Make sure that you only use LTS versions for the server installation: 22.04, 24.04, and so on. The semiannual interim versions are unsuitable for server installation as they have an update period of only nine months.

LTS

22.3.1 Installing Ubuntu Server

In 2017, Canonical changed the installer for the server installation. In the past an installation program originating from Debian was used, but now a new program called *Subiquity* with much simpler dialogs has replaced it. But as before, the installation program runs in text mode.

Subiquity

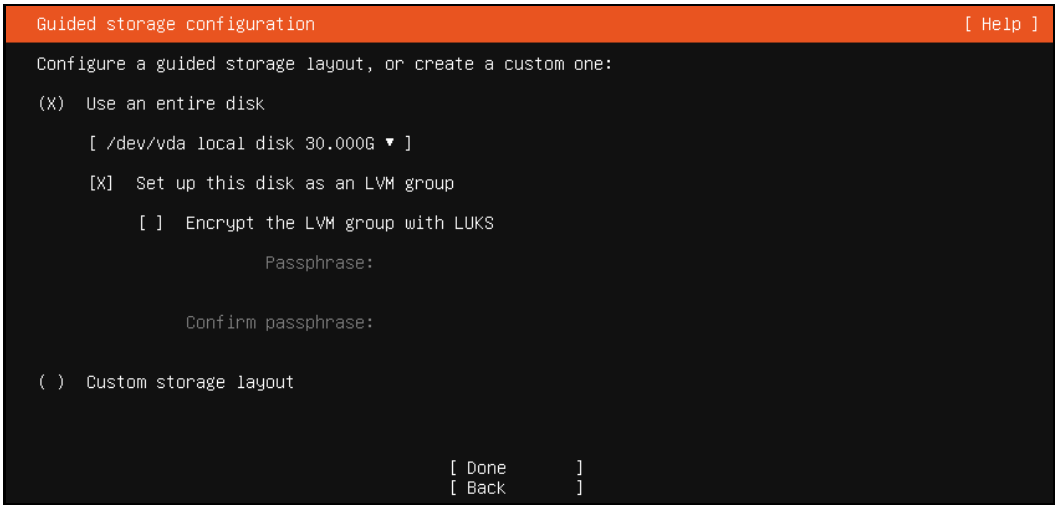


Figure 22.6 Multiple Dialogs in Text Mode to Guide Installation

In the first setup steps, you set the language, keyboard layout, and network connection. When configuring the file system, you have the choice of simply using the entire hard disk/SSD and setting up an LVM system there (see Figure 22.6). The installation program then creates a boot partition (2 GiB), an EFI partition (1 GiB) if necessary, and an LVM system with a logical volume for the root file system.

As an alternative, you can use the **Custom Storage Layout** option to partition one or more volumes manually and then set up RAID and LVM there if required (see Figure 22.7). Partitions serve as the basis for RAID and LVM, where you do not set up a file system and instead use the **Leave Unformatted** option.

Manual partitioning with RAID and LVM

Assuming you have a basic technical understanding of RAID and LVM, the operation of the dialogs is largely intuitive.

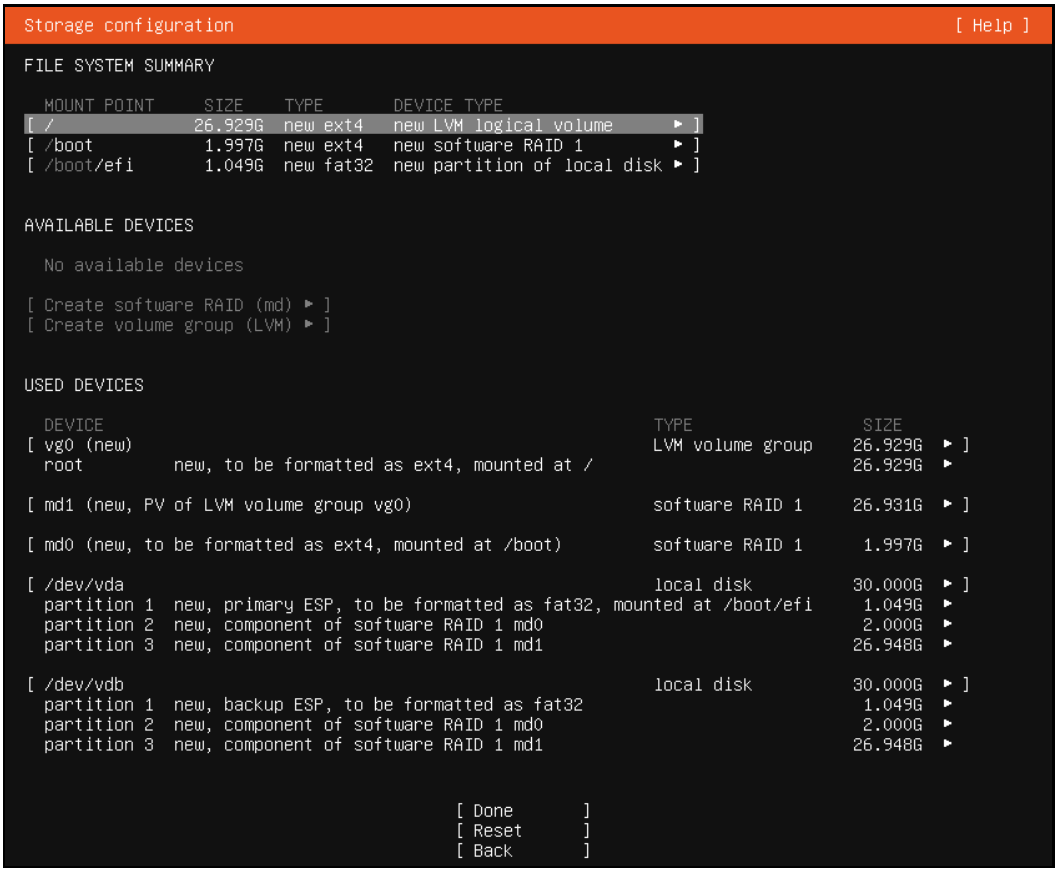


Figure 22.7 Manual RAID and LVM Configuration on Machine with EFI

An exception is the manual setup of the EFI partition. To do this, you need to select the volume (e.g., `/dev/vda`, not the **Free Space** entry below it) and run **Add as Boot Device**. Subiquity then reserves 1 GiB for a new partition, sets up a `vfat` file system there, and mounts it in the directory tree under `/boot/efi`. You cannot influence the size of the partition.

If you repeat the step for a second disk (**Add as Another Boot Device**), a corresponding partition is created there as well and initialized with boot files. However, the file system located there does not get integrated into the directory tree.

Make sure that you do not leave the dialog with Esc by mistake! If you do that, you will lose the entire configuration performed so far without warning!

Account and host name

While the setup program is starting the installation, you can enter your own name, password, and host name for the computer.

Software selection

In the dialog that follows, you usually choose the **Install OpenSSH Server** option. By default, login by password is provided. However, at this point you can also import a public part of your SSH key from GitHub or Launchpad and use it for authentication.

In addition, the setup program provides various server services as snap packages. I recommend that you skip this step and, if necessary, install your preferred services later in the form of traditional packages.

Finally, the installer performs a complete update of the distribution you just installed. After a restart, you can log in directly or via SSH and continue with the configuration.

22.4 Debian Server Installation

Unlike Ubuntu, Debian does not distinguish between a desktop and a server variant. There is only *one* Debian. For server installation, you want to use the default installer already described in Chapter 3, Section 3.1. You can do this either in graphics mode or in text mode. The dialogs displayed are identical in content but are a little easier to use with the mouse.

Partitioning

The only difference compared to a desktop installation concerns the often more complex partitioning of the disks, including RAID and LVM on a server. To be able to influence the partitioning yourself, you must select the **Manual** option in the first partitioning dialog. This takes you to a new dialog that contains some menu commands as well as a list of all existing hard disks or SSDs, partitions, and logical volumes.

Now select the **Free Space** entry in the partition table. If the HDD/SSD is brand-new, you must first initialize it—that is, set up a partition table.

In the dialog that opens next, choose the **Create a New Partition** option. Then specify the preferred partition size and how you want to use the partition—for example, as an EFI partition, for a common file system, or as a physical volume for RAID or LVM.

You can then join the partitions marked for **RAID** in a further step (**Configure Software RAID**) to form a group. Similarly, you can use individual partitions or RAID arrays as a data pool for a volume group. You then set up the required logical volumes and finally the file systems (see Figure 22.8). After defining all partitions, you need to run the **Finish Partitioning and Apply Changes** command in the partition table.

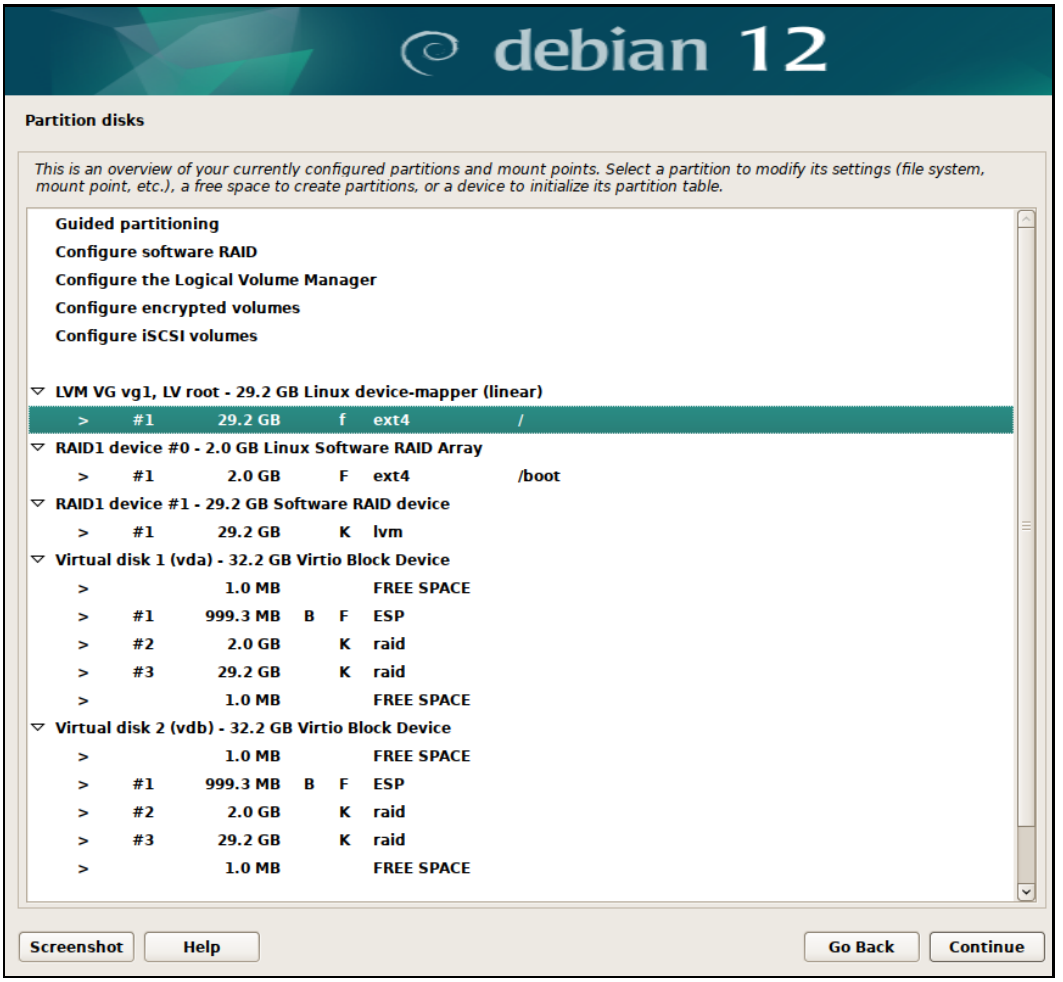


Figure 22.8 Manual RAID and LVM Configuration on Machine with EFI

If you're familiar with the structure of the partitioning dialogs, setting up RAID and LVM as well as formatting the file systems will succeed in a few minutes. Otherwise, it's recommended to practice the process first in a virtual machine with two or more disks.

Software selection

After installing the basic system and a few dialogs for configuring the package sources, you will be taken to the software selection. As a starting point for your own server, it's best to use a minimal system with an SSH server (see Figure 22.9). With an installation size of just under 2 GiB, you thus get a compact system.

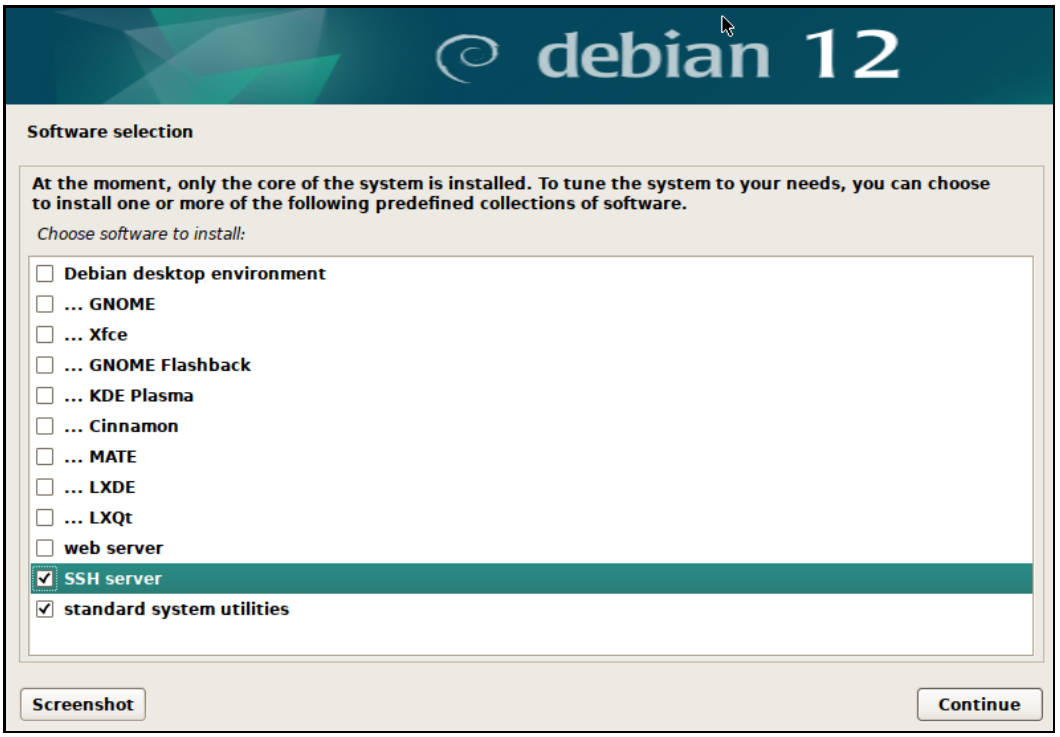


Figure 22.9 No Desktop System Required for Server Operation

22.5 Elastic Compute Cloud

Choose a cloud provider

There are countless providers that allow you to run virtual Linux machines in the cloud. This book is not the place for a market overview or even a vendor evaluation. Rather, I will describe Amazon's *Elastic Compute Cloud* (EC2) offering, which currently dominates the cloud market in its segment.

It should be mentioned that I have no contact with either company other than as a normal customer and, of course, I receive no compensation for presenting their offerings. This applies quite generally to all companies and products mentioned in this book, regardless of whether they are hardware, software, or services.

22.5.1 Amazon EC2

Outside the IT world, Amazon is primarily known as an internet-based department store. However, its *Amazon Web Services* (AWS) cloud offering generated around 14% of the company's revenue in 2022.

Elastic Compute Cloud

AWS is subdivided into countless services. In this section, I focus on EC2. As part of this service, you can rent computing power in the cloud and use it largely on demand. (Chapter 28, Section 28.10 briefly discusses another AWS service.) Provided you have an AWS login, you can use the following link as an entry point to EC2: *https://console.aws.amazon.com/ec2*.

Within EC2, the segmentation continues: there is a wide range of instance types. The RAM size, the architecture and number of CPU cores, the disk size, and the location of the data center vary. You can also choose between variants with or without SSD or NVMe or with or without GPU.

Abbreviations and Help Texts Galore

When you visit the EC2 website for the first time, you will be overwhelmed by abbreviations and strange names. In fact, the question sometimes arises as to whether all the terms and abstractions are really necessary to achieve the maximum flexibility that is always in the foreground, or whether the aim is rather to totally confuse the customer.

To Amazon's credit, the documentation provided on the AWS website is by and large excellent. The tutorials at the following web page are a good starting point: *https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/ec2-tutorials.html*.

When taking your first steps, you don't have to worry about doing something wrong and suddenly receiving a huge bill: Amazon provides with a relatively generous free quota, which is enough to try out at least the basic functions. The web interface even warns you about many settings that go beyond free usage.

22.5.2 Costs

There are many ways to create instances within EC2, and different prices apply to each variation. I have seldom seen such a confusing costing structure as with EC2. That said, this book is fortunately about technical details and not about costs. You will have to do your own research in this regard.

- *https://aws.amazon.com/ec2/pricing/* (CPU, RAM and network)
- *https://aws.amazon.com/ebs/pricing/* (I/O memory)
- *https://calculator.aws* (calculator)

However, to give you at least a kind of ballpark estimate, I have calculated the approximate costs per month for the following examples (as of spring 2024). The starting point is on-demand instances, which are recommended

for initial testing. Basically, you pay a certain amount for every hour your instance runs. This depends, among other things, on the computing power, the memory you have reserved, the server location, and the distribution (Ubuntu Pro is slightly more expensive than Ubuntu LTS because of licensing costs; the difference is significantly greater with RHEL):

- Ubuntu LTS, server location USA East (Ohio):
  - Instance type *t3.medium* (2 x86_64 CPU cores, 4 GiB RAM)
  - 100 GiB storage (EBS type *gp3*, universal SSD)
  - Instance cost about $40 per month
- Ubuntu Pro, otherwise as above:
  - Instance cost about $43 per month
- RHEL, otherwise as above:
  - Instance cost about $84 per month

Add to that, in both cases, approximately $90 per TiB of network traffic flowing from the instance to the internet. (The data is charged per GiB. In the opposite direction, traffic is free. If more than 10 TiB of data is used per month, the price per TiB decreases slightly.) In addition, there are costs for snapshots or backups in the S3 system.

On the other hand, instance costs decrease if you book computing power for a longer period (such as one year) in advance. Amazon refers to this as *Compute Saving Plans or EV2 Instance Saving Plans*. You can also save money when choosing the instance type. For the previous example, the *t3a.medium* type with an AMD CPU would also have been possible.

22.5.3 Getting Started

Of course, you'll need to register on the AWS website and provide the details of a valid credit card before you can start. Amazon recommends securing the login with two-factor authentication, but this isn't yet mandatory.

Setting up an SSH key

By default, SSH access to EC2 instances takes place via SSH with authentication by key. Amazon generates an SSH key pair along with the first instance if required. The public key remains with Amazon; you can download the private key once. From a safety point of view, this approach is not ideal: Who guarantees that the private key is not stored by Amazon after all (even if by mistake)?

That's why it's better if you upload the public part of your computer's SSH key, the `.ssh/id_rsa.pub` file, to Amazon on the following page: *https://console.aws.amazon.com/ec2/v2/#KeyPairs*.

This way you can definitely make sure that only you have the private key. In the unlikely event that you don't have such a key yet, you should generate one via `ssh-keygen` (see Chapter 23, Section 23.4).

Do Not Lose Your Key!

As in everyday life, it's also unfavorable when using the cloud if you lose your keys (files) or if they fall into the wrong hands if your notebook is stolen! In the case of EC2 instances, you then lose the ability to log into your virtual machines. For this reason, two basic safety rules apply: encrypt your notebook's hard disk/SSD when installing Linux, and make sure you have a backup of your keys. Ideally, the backup itself is also encrypted again.

22.5.4 Setting Up the First Instance

You usually start setting up a new instance by clicking the **Launch Instance** button at the following page: *https://console.aws.amazon.com/ec2/v2/home#instances*.

Which type of Linux?

The configuration is now done in several steps. In the first step, you have a choice of Windows, an Amazon-owned Linux distribution that I will briefly introduce ahead, Debian, RHEL, SUSE Enterprise Server, or Ubuntu Server.

The cost accounting for the enterprise distributions from Red Hat or SUSE is interesting. The license fees are already included in the EC2 fees. An instance with RHEL is therefore usually more expensive than one with Amazon Linux or Ubuntu Server. (However, you can even try out enterprise distributions as part of the free test quota.) If this selection seems slim, you can search the AWS Marketplace (*https://aws.amazon.com/marketplace*) for your preferred distribution. There you will find, for example, Amazon Machine Images (AMIs) of AlmaLinux, Kali Linux, Rocky Linux, and various other distributions. From this, you can also configure an EC2 instance. You only pay for the EC2 instance, not for Linux itself (unless you choose an enterprise distribution).

Custom Images

You can select one of your instances in the **Instances** dialog sheet and create your own image from it. This makes it easy to quickly create new, preconfigured instances for your own use.

Choose an instance type

The instance type defines the virtual base hardware of the virtual machine. Only the following types are currently available for free testing: t2.micro (x86 architecture, one core, 1 GiB RAM) and t4g.small (ARM, two cores, 2 GiB RAM). Although this does not sound very spectacular, it's sufficient for a simple, not too heavily frequented website. I have done my tests using t4g.small.

SSH key

In the key pair (login) step, you need to select a previously uploaded SSH key. This key is required for the first login.

Instance details and network configuration

In the next step, the web interface presents a whole collection of more or less obscure options. For initial tests, you won't go wrong if you simply keep all settings. If, on the other hand, you want to set up a large number of instances, you should take a closer look at the **Network** and **Subnet** setting boxes.

By default, each instance runs on a private network within the Amazon infrastructure (see Section 22.5.7). In AWS terminology, these networks are referred to as *virtual private clouds* (VPCs). When you launch multiple instances, they are usually in the same VPC and are separate from other customers' instances in other VPCs.

The **Network** setting allows you to specify in which VPC the instance will run. You also have the option of setting up your own VPCs and defining several subnets within them. These techniques are relevant when you want to run many instances but want to avoid having each instance communicate with every other instance. You can find background information about the configuration options at *https://docs.aws.amazon.com/vpc/latest/user-guide/how-it-works.html*.

Disks

Next, you must decide which disks your instance should be provided with. For small instance types, you get a volume (a virtual disk) from the Elastic Block Store (EBS) by default (this is how Amazon refers to its I/O infrastructure).

In addition to the size, the volume type is also available for selection here. By default, the **General-Purpose SSDs** variant is used (*gp2* type). Your instance is then allowed to perform 16,000 I/O operations per second, with a maximum throughput of 250 MiB/s. The free trial quota includes up to 30 GiB of this memory.

For instances with a high I/O load, you can choose the faster *gp3* variant or **Provisioned IOPS SSD** instead (*io1* or *io2* type), where you can freely set the preferred IOPS performance within a certain range. The following applies, of course: the more IOPS, the more expensive it becomes.

If required, you can take snapshots of your volumes in the AWS S3 storage for backup purposes. You can find a clear overview of the key EBS data in Wikipedia and comprehensive detailed information in the AWS documentation:

- *https://en.wikipedia.org/wiki/Amazon_Elastic_Block_Store*
- *https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/AmazonEBS.html*

For instance types with a high level of computing power, you can also book access to NVMe SSDs instead of EBS volumes. This gives you even more I/O performance, but you also have to pay more for it.

Linux instances typically use only a single partition that fills the entire disk and contains the root file system. The device name in the virtual machine is usually `/dev/xvda<n>`, `/dev/vda<n>`, or `/dev/nvme0p<n>`. (The quickest way to find this out is to run `lsblk` in the virtual machine.) One instance can be connected to multiple volumes. The first volume contains the root file system. You can partition the other volumes yourself and integrate them at any location in the directory tree.

A swap partition is not provided by default. You can set it up in a second volume if needed, or use a swap file instead. However, for performance reasons it's better to select an instance type with more RAM.

Tags

Instances, networks, volumes, and so on are automatically assigned very confusing ID numbers by AWS. You can add tags to help with the administration of your instances and other EC2 objects. These tags are key-value pairs with freely selectable strings. For example, it makes sense to save for each instance which distribution is running in it (e.g., key=OS, value=Ubuntu).

Security group

EC2 instances are secured from the outside by firewall rules. In the AWS terminology, these rules are formulated in *security groups*, and an instance can be associated with multiple security groups. By default, each instance receives a *launch-wizard-nnn* security group that blocks all incoming connection requests from the outside except SSH.

I explain further details of EC2's security concept and the associated configuration options in Section 22.5.7. The security group automatically generated by EC2 is appropriate for initial tests. In the longer term, it's usually advisable to define separate security groups for certain tasks (e.g., a web server).

Controlling

At the end of the page, the wizard displays a summary of all settings (see Figure 22.10). After a check or last corrections, you can launch your instance.

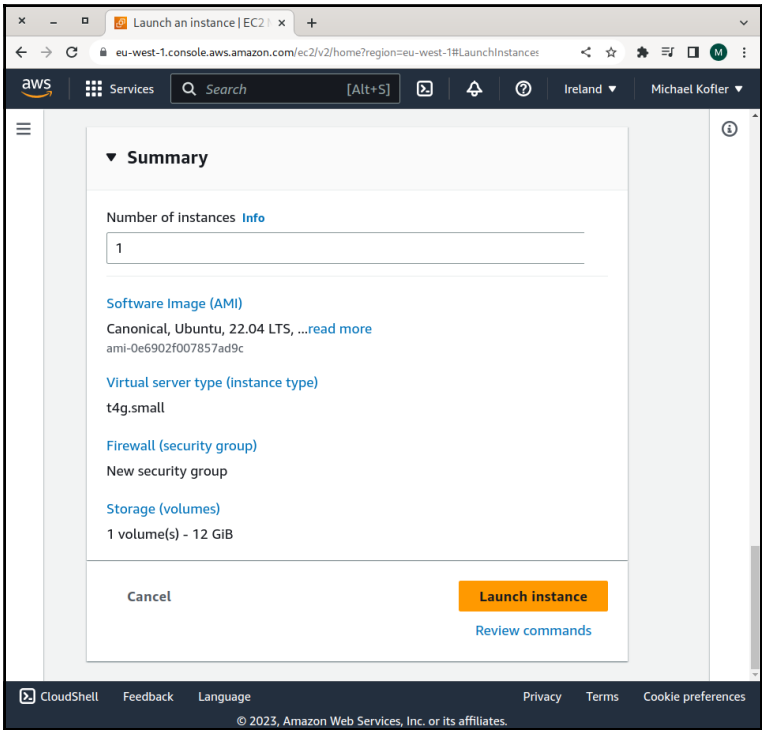


Figure 22.10 Summary of Key Data of New EC2 Instance

22.5.5 SSH Access

Once the instance is up and running, you probably want to log into the virtual machine via SSH and start configuring the server further. Provided that the instance is connected to the default key of your computer, you should run the `ssh` command as follows, replacing 1.2.3.4 with the IP address of the instance, of course:

```
user$ ssh ec2-user@1.2.3.4          (Amazon default)
user$ ssh ubuntu@1.2.3.4            (Ubuntu)
```

If the instance uses a different key, you want to specify it explicitly with the `-i` option:

```
user$ ssh -i .ssh/firsttest.pem ec2-user@1.2.3.4
```

Account name

Note that AWS usually provides the `ec2-user` account for SSH login, not `root`. In deviation from this, you must use the account name `ubuntu` for Ubuntu. If `ec2-user` doesn't work on other distributions or images, you will need to refer to their documentation to see which account is preconfigured.

sudo backup

As soon as the login succeeds, you can switch to the administrator mode by using `sudo -s` without a password (there is none yet!). `sudo` without a password is always a bit suspicious to me. To increase the security a little, you can set a password for `ec2-user` or the default account of your distribution and then change the `sudo` configuration in `/etc/sudoers` or in `/etc/sudoers.d/90-cloud-init-users` depending on the distribution:

```
# file /etc/sudoers oder /etc/sudoers.d/90-cloud-init-users
...
# comment out:
# ec2-user    ALL=(ALL)    NOPASSWD: ALL

# instead insert:
ec2-user    ALL=(ALL)    ALL
```

If No SSH Connection Can Be Established

If the `ssh` command fails, a security group that blocks all network traffic including SSH is probably the culprit. In the web console, check the security group settings for the instance and, if necessary, add a rule that allows SSH for *inbound* traffic from any source address (0.0.0.0/0).

22.5.6 EC2 Administration

The starting point for the administration of all EC2 instances is the **Instances** dialog sheet in the EC2 web console (see Figure 22.11).

There you can stop and restart your instances as needed. But be careful not to confuse **Instance State** • **Stop** and **Terminate**. **Stop** shuts down the instance, while **Terminate** deletes it for good! Also note that your instance will get a different IP address every time you restart it!

Changing the instance type

If the processing power of an instance is insufficient or if you feel that the instance is idle most of the time, then you can change the instance type via **Actions** • **Instance Settings** • **Change Instance Type**. The command is only available for selection if the instance has been shut down. There are also compatibility restrictions, which are described at *https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/ec2-instance-resize.html*. However, changing within a type family is no problem.

Increasing the data memory

In the **Volumes** dialog sheet, you can modify the properties of the volumes of your instances. Prior to that, it's recommended to name all volumes. By default, volumes are identified only by ID numbers. This makes it difficult to assign which volume belongs to which instance. The naming is most easily done in the volumes overview: if you move the mouse to the **Name** column, a text box for entering names will appear there.

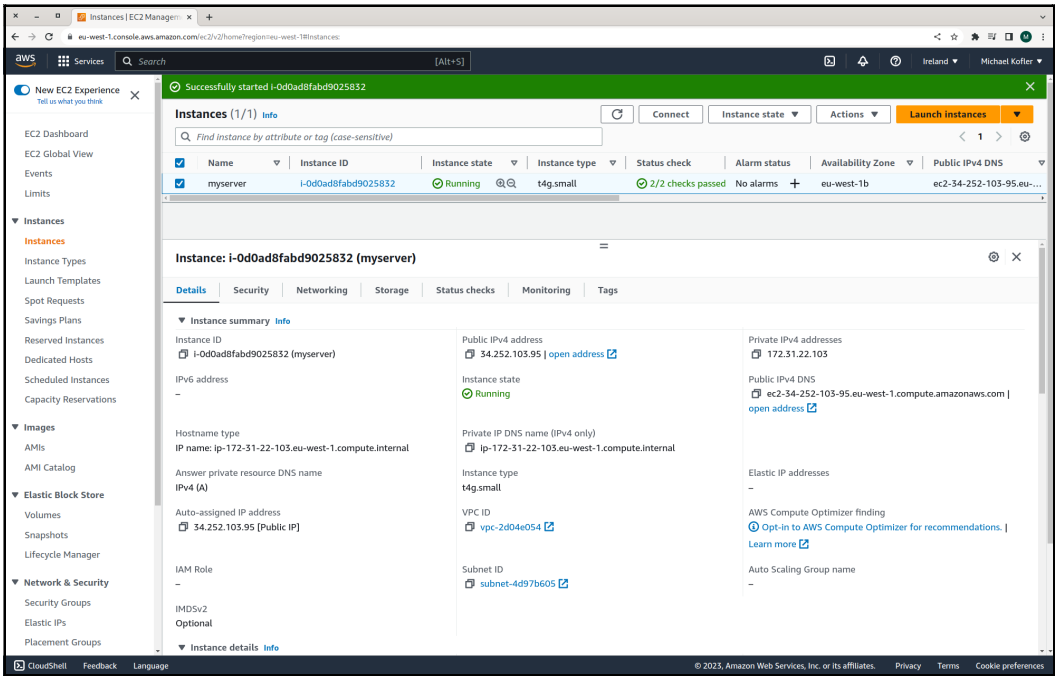


Figure 22.11 Overview of All Instances and Settings in Web Console

Probably the most important operation is enlarging a volume. To do this, select the volume and run **Actions • Modify Volume** (see Figure 22.12).

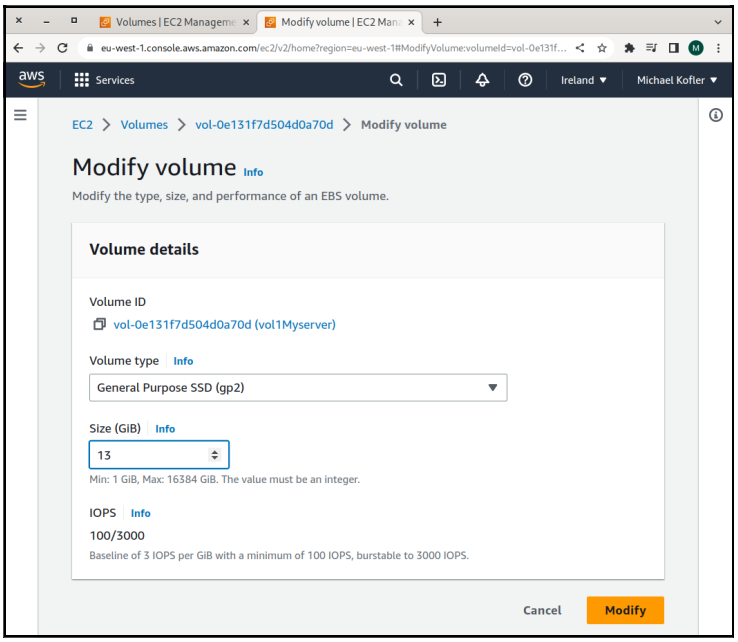


Figure 22.12 Changing Size of Volume

Note that you may increase volumes, but never decrease them. However, an enlargement during operation is possible.

After that, you need to make sure that your virtual machine uses the extra space. Sometimes this happens by itself, as if by magic; some distributions optimized for EC2 detect the enlargement of the virtual disk for the root file system and automatically adjust both the partition and the file system on it to the new size. You can convince yourself of this using the `lsblk` and `df` commands.

Depending on the distribution or if you are using multiple volumes, you may have to do it yourself and run commands such as `parted`, `resize2fs`, or `xfs_growfs` (see Chapter 18). The `growpart` command, which enlarges an existing partition as much as possible, is extremely helpful here. The command can be found in the `cloud-guest-utils` or `cloud-utils-growpart` package, depending on the distribution. The package is usually preinstalled. `growpart` works without problems in this example because the EFI partition (p15) is located *before* the system partition (p1), contrary to what the partition number suggests.

The starting point for the following commands was an original 12 GiB volume that was increased in size to 14 GiB. `lsblk` shows that the virtual `/dev/nvme0n1` disk is now 14 GiB in size, but the partition it contains still takes up only 11.9 GiB. When you call `parted`, this program offers to correct the partition table. `growpart` enlarges the partition, `resize2fs` then enlarges the `ext4` file system contained in it:

```
root# lsblk
NAME         MAJ:MIN  RM  SIZE RO TYPE MOUNTPOINT
...
nvme0n1      259:0    0    14G  0 disk
  nvme0n1p1  259:1    0  11.9G  0 part /
  nvme0n1p15 259:2    0    99M  0 part /boot/efi
root# parted /dev/nvme0n1
(parted) unit gib
(parted) print
Warning: Not all of the space available to /dev/nvme0n1 appears to be
used, you can fix the GPT to use all of the space (an extra 4194304
blocks) or continue with the current setting?
Fix/Ignore? fix
Number  Start    End      Size     File system  Name  Flags
15      0.00GiB  0.10GiB  0.10GiB  fat32              boot, esp
 1      0.10GiB  12.0GiB  11.9GiB  ext4
(parted) quit
root# growpart /dev/nvme0n1 1
root# lsblk
```

```
...
nvme0n1      259:0    0    14G  0 disk
  nvme0n1p1  259:1    0  13.9G  0 part /
  nvme0n1p15 259:2    0    99M  0 part /boot/efi
root# resize2fs /dev/nvme0n1p1
```

Deleting instances

There is no command in the EC2 web interface to delete instances. Instead, you should run the **Instance State • Terminate** command for the instance. This stops the instance and deletes the disks associated with it. The instance will remain visible in the web interface for a while but will no longer incur any costs. Eventually, the entry will disappear from the list of instances on its own. There is no command to explicitly trigger this operation.

22.5.7 Network Configuration

IPv4 addresses

By default, each EC2 instance receives a public IPv4 address at startup. It should be noted, however, that this address is only connected to your instance as long as the instance is running. When you stop your instance, the IP address returns into an address pool. At the next startup, the instance receives a new IP address. This is fine for first tests, but of course not for the permanent operation of a web or mail server.

There are two ways to use an IP address permanently:

- In the **Elastic IPs** dialog sheet, you can permanently reserve an IP address and then associate it with a running instance using **Actions • Associate Elastic IP Address**. This is possible during operation. Alternatively, you can assign the IP address to a new instance at startup.

  As long as the instance is running, this will not incur any additional costs. However, if you stop the instance and no longer need it, you must remember to release the elastic IP address. To do this, you must disassociate the IP address from the instance (**Disassociate Address**) and return it to Amazon, so to speak (**Release Address**). Otherwise, you will be charged for the unused IPv4 address that you have blocked (see *https://aws.amazon.com/ec2/pricing/on-demand/#Elastic_IP_Addresses*).
- Large companies can transfer part of their own IPv4 network to Amazon to use the addresses there for their own instances (see *https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/ec2-byoip.html*).

Regardless of how you connect the instance to a public IP address, internally, the virtual machine is always located in a private local Amazon network. `ip addr` displays this private address, even if EC2 has connected your instance to a public IP address on the outside. You can only determine the actual address of your instance in the AWS web interface:

```
root# ip addr show ens5
2: ens5: <BROADCAST,MULTICAST,UP,LOWER_
UP> mtu 9001 qdisc mq state UP ...
    link/ether 06:c4:8e:e1:92:05 brd ff:ff:ff:ff:ff:ff
    inet 172.31.22.103/
20 metric 100 brd 172.31.31.255 scope global ...
    ...
```

Reverse DNS If you run a mail server in the instance and need a reverse DNS record (see Chapter 26, Section 26.1), you must select the elastic IP address and run **Actions • Update Reverse DNS**.

IPv6 Oddly enough, new instances do not get an IPv6 address by default. EC2 of course supports IPv6, but the required configuration of the network assigned to your instance (VPC; see the next paragraph) is very complex. You can find relevant instructions at *https://docs.aws.amazon.com/vpc/latest/userguide/vpc-migrate-ipv6.html*.

Firewall By default, your EC2 instances are located in a kind of local network (VPC) and can communicate with each other in it without any restrictions. To the outside world (i.e., the internet), they are protected by an internal AWS firewall.

By default, EC2 creates a new security group (named *launch-wizard-<nnn>*) when setting up an instance, which only allows traffic through port 22. This allows you to administrate your instance via SSH. Apart from that, any traffic coming from the outside will be blocked. This is even true for ping packets. If necessary, you can define exception rules for this group, such as to allow HTTP and HTTPS (see Figure 22.13).

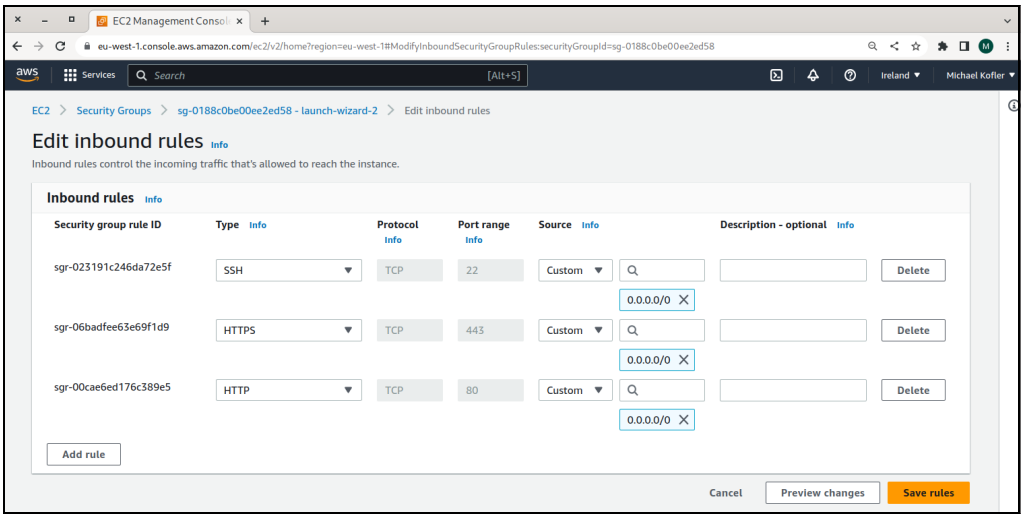


Figure 22.13 Setting Firewall Rules for Security Group of Instance

Alternatively, you can assign the `default` security group to your instance during setup or later. However, this leaves the virtual machine completely unprotected.

22.5.8 Amazon Linux

By default, AWS suggests that you use Amazon Linux. Does that mean Amazon has now developed its own distribution? Not quite. The current Amazon Linux 2023 is a Fedora clone with a maintenance period of five years (including two years of standard support and three more years of reduced maintenance support). Amazon plans to ship a new version every two years. So the next version is expected to be Amazon Linux 2025. Unlike Fedora, Amazon Linux includes a live kernel patching feature. For more information, visit the following pages:

- *https://aws.amazon.com/linux/amazon-linux-2023/features*
- *https://aws.amazon.com/blogs/aws/amazon-linux-2023-a-cloud-optimized-linux-distribution-with-long-term-support*

Firewall

I've mentioned before that security groups are used to define firewall rules for the instance at EC2 level. In this respect, another firewall *inside* the virtual machine is no longer useful. Amazon Linux takes this into account: the FirewallD system, which is active by default in Fedora, is not installed at all in Amazon Linux.

22.5.9 Internal Details

CLI

I have so far assumed that you manage your EC2 instances via the AWS web interface. For beginners, this is undoubtedly the best way. However, if the number of your instances is growing and you want to administrate them using scripts, you should familiarize yourself with the `aws` command.

The installation and startup of this command is described in Chapter 28, Section 28.10, where you assign the `AmazonEC2FullAccess` policy to the group that is assigned to the user. As a first test, you can use the following command to determine detailed information about your instances. The output is in JSON format and is unfortunately difficult to read:

```
user$ aws ec2 describe-instances
{
    "Reservations": [
        {
            "Groups": [],
...
```

You can find a reference to the numerous control options at *https://docs.aws.amazon.com/cli/latest/reference/ec2*.

Hypervisor

Behind the scenes, AWS used to rely on Xen as its virtualization system. In 2017, it became known that Amazon had started to migrate at least parts of its cloud infrastructure to KVM. However, Amazon doesn't really want to allow anyone to look behind the scenes. I have found little specific information about which instance types use which hypervisor.

Once an instance is running, you can determine the virtualization system being used quite easily by using the following command:

```
root# dmesg | egrep -i 'virtual|vbox|xen|kvm|hypervisor'
Hypervisor detected: Xen HVM
...
```

I have tried the command on two test instances. One was run with KVM, the other with Xen.

Chapter 23
Secure Shell (SSH)

Servers that are not physically in front of you can only be administrated via a network connection. The preferred tool for this is SSH (*Secure Shell*). It enables both the simple execution of commands and operation of graphical programs over a secure network connection. The only requirement is that an SSH server is installed on the remote computer.

The `ssh` command is a secure successor to `telnet` and `rlogin`. The client use of this command has already been described in Chapter 11, Section 11.2. This short chapter provides some tips on how to set up an SSH server as securely as possible. The chapter also discusses the use of keys for authentication.

23.1 Installation

In many distributions, the SSH server already moves to the hard disk or SSD during the initial installation. Only if the SSH server is not yet installed do you have to do it yourself.

Debian, Ubuntu

On Debian and Ubuntu, you must install the `openssh-server` package. The `sshd` daemon is automatically started immediately:

```
root# apt install openssh-server
```

Fedora, RHEL

On Fedora and RHEL, you must perform the installation in a similar way using `dnf` or `yum`:

```
root# dnf install openssh-server      (Fedora, RHEL)
```

On Fedora, the SSH server is automatically executed immediately. On RHEL, on the other hand, both the initial startup and the activation of the automatic startup must be performed manually:

```
root# systemctl enable --now sshd     (start SSH server)
```

SUSE

On SUSE, the package name is `openssh`. The package is installed by default. During the installation, however, there are two options that control whether the SSH server is actually started and whether the firewall should block port 22. The default settings are *do not start* and *block*. So, if you overlooked these options during installation, you should allow the SSH service now in the YaST **Firewall** module and also run the following command:

```
root# systemctl enable --now sshd     (start SSH server)
```

Firewall The access to the SSH server fails if a firewall blocks port 22. By default, this happens only on SUSE.

23.2 Configuration and Security

The configuration files for `sshd` are located in the `/etc/ssh` directory. For the server configuration, `sshd_config` is responsible. (Do not confuse the file with `ssh_config` for the client configuration!)

Usually, `sshd_config` can be left unchanged; the SSH server should work right away. By default, the communication takes place via IP port 22 by default. If you have special requirements, you will find a lot more information in the `man` pages.

Secure FTP A component of the SSH server is the `sftp` server. This is a secure alternative to a regular FTP server. The `sftp` functions are usually available automatically as soon as the SSH server is running. If necessary, you should take a look at the `Subsystem sftp` line in `sshd_config`. The `sftp` functions can be used with `sftp`-compatible clients, such as the `sftp` program.

Securing An active SSH server is a security risk that should not be underestimated. Any hacker who guesses a valid username and password combination can log into your computer! Attackers use automated tools that search the internet for servers and try to log in. All such activities are recorded in the `/var/log/auth.log` or `/var/log/secure` file or journal, depending on the distribution. If you run a publicly accessible server, you will find thousands of login attempts there every day.

The following measures each reduce the likelihood of a crack attack on your SSH server. They can be used individually or in combination.

No root login An attacker wants to obtain `root` privileges—and the easiest way to do that is, of course, through a `root` login. Only one parameter (the `root` password) needs to be guessed. It is therefore much safer to prohibit direct `root` login via SSH. For administration, you must therefore log in under a different username and then switch to the `root` mode via `sudo`. If you administrate a root server, be sure to test this before making the following change, as otherwise you may lock yourself out:

```
# change in /etc/ssh/sshd_config
...
PermitRootLogin no
```

For the change to take effect, you must prompt `sshd` to reread the configuration files:

```
root# systemctl reload sshd
```

For the attacker, this has the consequence that two parameters are now unknown: the login name *and* the password!

A reasonable alternative to `PermitRootLogin = no` is the `prohibit-password` setting or the equivalent `without-password` keyword:

```
# change in /etc/ssh/sshd_config
...
PermitRootLogin without-password
```

Don't worry, this setting by no means allows a `root` login with an empty password! `without-password` rather means that a login is still possible, but that the authentication must be done by a more secure method than the simple password entry—usually by exchanging keys (see Section 23.4).

Locking the password authentication

`PermitRootLogin` applies only to `root`. If you want to generally exclude any authentication by password, you should include the `PasswordAuthentication no` statement in `sshd_config`. Authentication is then possible exclusively via keys. Don't forget to test this beforehand; otherwise, you might block the only network access to your server. Also note that you can now only log into your server if you have your notebook with your key file. You definitely need a backup of your private key in case your notebook is stolen!

Blocking IPv6 access

If the server has both a public IPv4 address and an IPv6 address, it may be appropriate to block IPv6 access to the SSH server. Here's why: password cracking attacks via IPv6 can hardly be prevented by Fail2Ban (see the following section).

You can set the desired IP variant using `AddressFamily`. `inet` means that only IPv4 is allowed. Other allowed settings are `inet6` (IPv6 only) and `any` (both IPv4 and IPv6, applies by default):

```
# change in /etc/ssh/sshd_config, accept IPv4 only
...
AddressFamily inet
```

Changing the SSH port

By default, the SSH server communicates via port 22. The `port` line enables you to easily set another, currently unused port. Because many automated crack tools only take port 22 into account, you can avoid many security problems at a single stroke.

On Fedora and RHEL systems, you must also notify SELinux of the port number change:

```
root# semanage port -a -t ssh_port_t -p tcp <newportnr>
```

When using `ssh`, you must then explicitly specify the port of your SSH server each time with `-p`. Note that you must use the `-p` option with the `scp` command; `-p` here means *preserve* and ensures that the time and access information of the file to be copied will be preserved!

However, you should be aware that the protection provided by the port change has only a limited effect. Anyone who seriously wants to attack your server and is not just looking for the next best poorly configured server to install a root kit will perform a port scan. This means that your SSH server will not remain undetected for long, no matter what port it is running on.

Changing the SSH port also has a significant disadvantage. While most firewalls are configured to allow traffic through port 22, this will likely not be true for your new port. If you want to quickly access your server via SSH from a company where you just work for a few days, you may already fail at the company firewall.

23.3 Fail2Ban

SSH is an inherent security risk in that a suitably persistent attacker can sooner or later guess a valid login password through endless trial and error. Hackers can use automated password cracking tools such as `hydra`, `ncrack`, or `medusa` to do this.

Fortunately, it's relatively easy to prevent such attacks by simply blocking a particular host for a while after several unsuccessful attempts. This is exactly what the Fail2Ban program does. It can monitor various other network services besides SSH. The attacking computer is blocked by firewall rules.

DenyHosts

In the past, the simpler DenyHosts program was often used instead of Fail2Ban. However, it is no longer maintained and is no longer available in the package sources of many distributions.

Installation

On Debian and Ubuntu, the `fail2ban` package is available in the normal package sources. On RHEL, the package is missing, but it is included in the EPEL package source.

Configuration

After installing the `fail2ban` package, you need to create a copy of the default `jail.conf` configuration file in the `/etc/fail2ban` directory, and name the copy `jail.local`. That's where you will make all further changes.

These changes then take precedence over the basic settings in `jail.conf`. In addition, `jail.local` is prevented from being overwritten during package updates:

```
root# cd /etc/fail2ban
root# cp jail.conf jail.local
```

Fail2Ban refers to a service to be monitored as a *jail*. The supplied configuration files in `/etc/fail2ban` contain rules for all possible services (jails); however, by default they are all disabled. To enable SSH rules, you need to look for the `[sshd]` section in `jail.local`. In this section, you must add the `enabled = true` line:

```
# file /etc/fail2ban/jail.local
...
[sshd]
port    = ssh
logpath = %(sshd_log)s
enabled = true
```

The default configuration of Fail2Ban is to lock a host for 10 minutes after five unsuccessful SSH login attempts within 10 minutes. The three relevant parameters—`maxretry`, `bantime`, and `findtime`—are predefined together for all services in `jail.local`. You can easily change these default values if necessary or specify other values in the `[sshd]` block for SSH only. Details about the configuration of Fail2Ban can be found in `man jail.conf`.

Now you just have to make sure that Fail2Ban is started: **Start**

```
root# systemctl restart fail2ban
root# systemctl enable fail2ban
```

You can determine the current status of Fail2Ban using the following command: **Status**

```
root# fail2ban-client status sshd
Status for the jail: sshd
  filter
     File list: /var/log/auth.log
     Currently failed:  15
     Total failed:      1416498
  action
     Currently banned:  9
     Total banned:      216628
     Banned IP list:    177.n.n.n  43.n.n.n ...
```

A log with all IP addresses that are currently blocked or have been blocked in the past is provided by the `/var/log/fail2ban.log` file.

IPv6

In general, Fail2Ban is IPv6-compatible. However, as there are almost infinite IPv6 addresses, an attacker can use a different IPv6 address for each attempt. In this respect, blocking individual IPv6 addresses does not provide any real protection.

sshguard

An alternative to Fail2Ban is the *sshguard* program (see *https://www.sshguard.net*).

23.4 Authentication with Keys

Generating keys

The operation of an SSH server is more secure if authentication can only be carried out using digital keys instead of passwords. For this purpose, you first need a local key pair consisting of a private and a public key file. You can create these two files by running `ssh-keygen`. If the command detects an already existing key, it warns you against overwriting it by mistake. `ssh-keygen` offers to encrypt the key itself by means of a *passphrase* (a password). For security reasons, this is recommended. If your private key falls into the wrong hands, the thief cannot do anything with the key without a password.

If you simply answer the passphrase question by pressing Enter or by typing "empty", `ssh-keygen` will forgo the encryption. This is convenient because it allows the usage of SSH without a password prompt. However, you are taking a safety risk by doing so! Anyone who gets hold of your key on the client computer can easily log on to all computers on which you have installed the public part of the key:

```
user@client$ ssh-keygen
Generating public/private rsa key pair.

Enter file in which to save the key (/home/user/.ssh/id_rsa):
<Return>
Enter passphrase (empty for no passphrase): ********
Enter same passphrase again: ********
Your identification has been saved in /home/user/.ssh/id_rsa.

Your public key has been saved in /home/user/.ssh/id_rsa.pub.
```

Key files

A key pair consists of two files that are usually named `id_rsa` and `id_rsa.pub`. The files are stored in the `.ssh` directory. You can pass on the public part of the key, such as by storing it on another SSH server via `ssh-copy-id` (discussed ahead) or uploading it to the cloud provider before setting up

a cloud instance. The private key, on the other hand, remains with you; you must *never* give this file to anyone. (However, you should have a backup of the key pair in a safe place.)

Two-part keys are a great cryptographic invention, but they do not correspond to our human imagination. The following analogy fits better:

- The private SSH key is equivalent to a conventional key.
- The public SSH key, on the other hand, corresponds to a lock.

Thus, you have an unlimited supply of locks (copies of the public key) that you can place anywhere you can think of, especially on foreign servers. However, you never give the corresponding key—the private key file—out of your hand.

Many SSH installations use the RSA algorithm by default. The key pair then consists of the previously mentioned `id_rsa` and `id_rsa.pub` files. Newer SSH versions, on the other hand, use the even more secure Elliptic Curve Digital Signature Algorithm (ECDSA). In this case, the key files have different names (see Table 23.1).

	Private Key	Public Key
RSA	`id_rsa`	`id_rsa.pub`
ECDSA	`id_ecdsa`	`id_ecdsa.pub`
Curve25519	`id_ed25519`	`id_ed25519.pub`

Table 23.1 Names of Key Files Depending on Algorithm

Uploading the key to the server (ssh-copy-id)

Once you have created a key pair, you must transfer the public part to the SSH server using the `ssh-copy-id` command. Your key will be installed in your home directory on the server in the `.ssh/authorized_keys` file. During the execution of `ssh-copy-id`, you have to authenticate one last time with your password:

```
user@client$ ssh-copy-id -i user@server
user@server's password: *******
```

The `ssh-copy-id` command fails if the server only allows the authentication by key. To solve this chicken-and-egg problem, you must perform the key transfer from another client that can authenticate at the server. In this case, you must transfer your public key file, *.ssh/id_rsa.pub*, to the server via a third computer and manually append it to the end of the `.ssh/authorized_keys` file there.

The next login

As soon as the key transport has succeeded, `ssh` automatically matches the key pair on the next login attempt. `ssh` thus checks whether the public key stored on the server matches the private part on your local computer. This process happens automatically. Logging in with your server password is now no longer required. However, you must provide the passphrase of your private key file if you have secured your key in such a way:

```
user@client$ ssh user@server
Enter passphrase for /home/user/.ssh/id_rsa: ******
```

ssh-agent

If your keys are protected by a password or passphrase, you have gained security by using keys, but not in terms of convenience. A secure yet reasonably convenient solution is provided by `ssh-agent`. This program manages all of a user's private keys. The program can be started as follows:

```
user$ eval $(ssh-agent)
```

This changes some environment variables of the current console. `ssh-agent` continues to run as a background process. Through `ssh-add`, you can now add your private keys:

```
user$ ssh-add ~/.ssh/id_rsa
Enter passphrase for /home/user/.ssh/id_rsa: ******
```

From now on, `ssh` uses the key files managed by `ssh-agent`. This means you will never be asked for the password for the key file again. Instead of entering the password with every `ssh` command, the entry is only required once. The major disadvantage of `ssh-agent` is that the effect of the environment variables is limited to a single console.

gnome-keyring

On GNOME, `gnome-keyring-daemon` takes care of passwords and SSH keys by default. The first access to the data usually requires a correct login (avoid autologins from the display manager!) or the specification of the master password. To administrate the stored keys and passwords, you can use the `seahorse` program.

Login problems

The password-free login to the SSH server does not always work straight away. This is due to various security settings. If `sshd_config` contains the `StrictModes yes` setting, the `authorized_keys` file will only be included if its access rights are set very restrictively. If necessary, you must run the following two commands:

```
user@server$ chmod 700 ~/.ssh
user@server$ chmod 600 ~/.ssh/authorized_keys
```

Fedora and RHEL servers sometimes experience another problem: the key file generated by `ssh-copy-id` lacks SELinux context information. In this case, you can fix the situation as follows:

```
root@server# /sbin/restorecon -r /root/.ssh          (for root)
root@server# /sbin/restorecon -r /home/user/.ssh     (for other users)
```

Troubleshooting

If the SSH login doesn't work as expected, you should run the `ssh` command with the additional `-v` option. `ssh` will then return numerous debugging messages. The option can be specified up to three times—that is, with `-v -v -v`. `ssh` then logs in more detail accordingly.

Turning off password login altogether

Once establishing an SSH connection with the key works and no login prompt appears, you can consider disabling the password login altogether. To do this, you need to modify the `sshd_config` configuration file on the server: Two lines are crucial:

```
# in /etc/ssh/sshd_config
...
PasswordAuthentication no
UsePAM                 no
```

This means that from now on SSH authentication is *only* possible with keys. Be careful not to lock yourself out of your system, though! If you lose the key on your client computer, perhaps because your notebook is stolen and you have no backup, you can no longer log in on the server! As usual, you pay for any additional security with less flexibility.

Multiple keys

Until now, I have assumed that you only use a single, self-generated key. But maybe you have multiple keys that you have used on different machines in the past, or you want to use different keys for different external servers, or someone hands you a key file so you can administrate another machine.

The simplest solution is to set up the `config` file in the `.ssh` directory on the client machine. Do not forget `chmod 600`! This file contains references to all key files with the `IdentityFile` keyword—for example, like this:

```
# .ssh/config file
IdentityFile ~/.ssh/id_rsa
IdentityFile ~/.ssh/id_rsa.my-other-server
IdentityFile ~/.ssh/id_rsa.kvm-host
...
```

In this configuration, the `ssh` command simply tests all keys until one of them matches. You can add even more "intelligence" to the `config` file by grouping the entries by hosts. This allows you to additionally specify for each host which port, user, and so on should be used. The following lines illustrate the syntax; more details can be found in `man ssh_config`:

```
# .ssh/config file
Host kofler.info
  IdentityFile ~/.ssh/id_rsa
  User top secret
Host my-other-server.com
  IdentityFile ~/.ssh/id_rsa.my-other-server
  User otheradmin
  Port 22022
```

23.5 Two-Factor Authentication

When you use SSH, two authentication methods are common: a simple password (flexible, but not perfect in terms of security) or the use of keys as described previously (more secure, but requires that you always have a notebook with your private key at hand). A third option is to set up two-factor authentication.

In addition to the first factor (either a conventional password or a key), a second piece of information must then be passed during login. This can be, for example, a one-time key generated by an app on a smartphone or a key generated by an external token (such as the YubiKey).

With two-factor authentication (2FA), you increase the level of security. Unfortunately, this also makes logging in much more tedious. You also need to make provisions for the emergency of losing the second factor. These are all reasons that 2FA has not yet been able to gain widespread acceptance.

23.5.1 2FA with Google Authenticator

Idea

Google Authenticator is an app that can be installed on Android and Apple smartphones. Once an account has been set up there, the app generates a new six-digit key every 30 seconds, which can be used as a second factor when you log in. This app works purely locally. Neither the keys nor the one-time codes are transferred to a Google server. The appeal of this solution is that the second factor is a device that you normally always have with you anyway: your smartphone.

The algorithms used to generate the codes are publicly available as open-source code. That's why you can also use other apps instead of Google Authenticator. Personally, I find Authy especially attractive. I will briefly introduce this app at the end of the section.

Setting up Google authentication

On the Linux server to be secured, you must install `google-authenticator` or `libpam-google-authenticator` package, depending on the distribution. It contains a PAM module to control one-time codes and the `google-authenticator` command to set up the authentication feature. You need to run this command for the account you want to log in to:

```
user$ google-authenticator
  Do you want authentication tokens to be time-based (y/n)
  Do you want me to update your .google_authenticator file? (y/n)
  Do you want to disallow multiple uses of the same
    authentication token? (y/n)
  ...
```

You can answer all queries with `y`. The queries are omitted if you execute the command with the `-t -d -f -r 3 -R 30 -W` options. The command displays a QR code (see Figure 23.1) that you must scan using the Google Authenticator app on your smartphone.



Figure 23.1 Setting Up Google Authenticator

This is how you set up a new account in the smartphone app. This tells the app about your account, and it immediately starts generating six-digit one-time codes. Each code is valid for 30 seconds. After that, the next code is generated automatically.

The One-Time Codes Are User-Specific!

Note that each account is valid only for a combination of hostname and user! So if you run `google-authenticator` as user `as34`, for example, you cannot use the codes generated in the authenticator app to log in on the same machine as `bf72` or as `root`!

Emergency codes

The `google-authenticator` command also issues five *emergency scratch codes*. These are also one-time codes, but without a time limit. If you don't have your smartphone at hand, you can use one of these codes to log in (but only once!). Before the five codes are used up, you should add new codes to the `.google_authenticator` file using an editor. (It isn't necessary to generate the codes with a program. Each code you enter here is valid exactly once.)

PAM configuration

For Google Authenticator to be used, you need to modify two configuration files. The following lines relate to the PAMs for the SSH server:

```
# file /etc/pam.d/sshd
# force 2FA for all accounts
# add this line at the beginning of the file
auth required pam_google_authenticator.so
```

Note that these configuration changes apply to *all* accounts on the respective server! Users for whom Google Authenticator has not been set up previously will no longer be able to log in. (Adding new users also becomes much more cumbersome!) Alternatively, if you want 2FA to apply only to accounts where `google-authenticator` has been set up, you can use the following configuration. Needless to say, this will weaken the protection provided by 2FA:

```
# file /etc/pam.d/sshd
# 2FA only for accounts where .google_authenticator exists
auth required pam_google_authenticator.so nullok
```

SSHD configuration

You also need to modify the configuration file of the SSH server. The following four settings are required, whereby some options are usually already correctly preset:

```
# File /etc/ssh/sshd_config
# Change existing setting
UsePAM                          yes
ChallengeResponseAuthentication  yes
AuthenticationMethods            keyboard-interactive
```

Finally, you need to use `systemctl restart sshd` to restart the SSH server.

Warning

Try the procedure in a virtual machine first, not on a real server! If you set up Google Authenticator for a server to which you do not have physical access, you should always remain logged into a separate SSH session with `root` privileges during the entire process until a successful test has been completed. This allows you to carry out repair work via this connection if necessary. Otherwise, you may not be able to log in to your own server if anything goes wrong during the configuration process!

The additional security comes at the price of a new dependency: if you have left your smartphone somewhere or the battery is empty, then the SSH login will only succeed with one of the emergency codes.

Usage

A login to the server now looks as follows:

```
user$ ssh user@hostname
Password: ******              (the regular login password)
Verification code: ******     (displayed on the smartphone)
```

The login now always requires two passwords. Unfortunately, a login with a previously set up key (`ssh-copy-id`) plus security code does not work.

What happens if you lose your smartphone? In the past, you could not create a backup of your 2FA accounts using the Google Authenticator app. In March 2023, Google finally introduced a corresponding function, but it was associated with considerable restrictions at that time.

Authy

The free Authy app (*https://authy.com*) does a much better job in this respect. It generates the same codes as the Google Authenticator app and also provides a lot of useful additional features, including a synchronization of the 2FA accounts set up across multiple devices. For this to work, the account data must be stored on a server of the Twilio company. Even if your data is encrypted with a password of your choice (end-to-end encryption), any external storage is of course a certain security risk.

23.5.2 2FA with YubiKey

Instead of codes generated by a smartphone, you can use a *security token*, which looks like a USB flash drive, as a second factor for SSH login. Among other functions, the device simulates keyboard input as soon as you press a touch-sensitive button. (The computer to which the security token is connected regards the token as a USB keyboard.) The string that is intended as a one-time password (OTP) consists of two parts: The first 12 characters always remain the same and are, in a sense, the public part of the key. The remaining characters are the actual password, which changes every time.

The string is generated based on a nonreadable key. Together with a symmetric key, which you can determine on the token manufacturer's website, it's possible to check whether a string matches your token.

We conducted our tests using the YubiKey 5 NFC model from Yubico. Comparable devices are also available from other providers, including Google (Titan Security Key).

API key

To enable a verification of your YubiKey one-time passwords, you will need to generate a key at *https://upgrade.yubico.com/getapikey*. For this purpose, you must enter your email address and fill in another input field by touching the YubiKey. The website responds with an ID and the *API key*. After that, you will need both data elements for the configuration of the YubiKey PAM module.

PAM module

On the machine running the SSH server, you must install the `yubico` PAM module. For Ubuntu, Yubico provides a package source:

```
root# add-apt-repository ppa:yubico/stable
root# apt update
root# apt install libpam-yubico
```

For RHEL, there's a suitable package in the EPEL package source:

```
root# dnf install pam_yubico
```

For other distributions, you can find instructions on how to download the source code from GitHub and compile the module yourself at *https://developers.yubico.com/yubico-pam*.

PAM configuration

To make sure the PAM module will be used, you must add the following statement to the end of `/etc/pam.d/sshd` in a single line and without the `\` character:

```
# at the end of /etc/pam.d/sshd
auth required pam_yubico.so id=12345 key=apikey \
  authfile=/etc/yubikey-mappings mode=client
```

Here, you replace `12345` with your ID and `apikey` with your API key. Both data elements originate from the website described previously.

Mapping file

For all users whose login is to be checked via YubiKey, the Linux account name and the first 12 characters of the one-time password must be entered in a mapping file. You specify the location of this file during the PAM configuration:

```
# File /etc/yubikey-mappings
michael:ccccnixgfask
peter:ccgsdkgalfja
...
```

Make sure that you really specify exactly 12 characters and that you do not include any spaces before and after the colon! If you have multiple YubiKeys that you use, the syntax is `name:key1:key2:key3` and so on.

sshd configuration

Finally, you only need to adjust the configuration of the SSH server so that the new authentication procedure will actually be used. You can do this in a similar way to using Google Authenticator. In the following listing, however, 2FA is not generally activated, but only for selected accounts:

```
# /etc/ssh/sshd_config
# Change existing setting
UsePAM                          yes
ChallengeResponseAuthentication  yes

# add at the end
Match User michael,peter
  AuthenticationMethods keyboard-interactive
```

Before you activate the changes via `systemctl reload sshd` and then try them out, you should make sure that an active SSH connection to the server is always maintained. Otherwise, you'll run the risk of locking yourself out in the event of a configuration error.

No Login without a Yubico Server

I would like to make one more point here: every time you log in, `pam_yubico` contacts a Yubico server and checks whether the OTP you have entered actually matches your token. At the same time, the test prevents an OTP from being used more than once. You can find more technical background information on the procedure at *https://developers.yubico.com/OTP/OTPs_Explained.html*.

Yubico currently operates five OTP servers that are distributed around the world to ensure a relatively high level of redundancy. However, if these

servers suddenly become unavailable due to a technical glitch, hacking attack, or network problems, you will no longer be able to log on. In this respect, it's a good idea not to activate 2FA for all accounts on a server and to leave a less secure emergency account with a "normal" login.

Test An SSH login to your server should now work as follows:

```
user$ ssh peter@a-company.de
Password: **********                (regular password, input via keyboard)
YubiKey for 'peter':  ***********   (OTP, input by touching the YubiKey)
```

YubiKey on Linux The use of YubiKeys as well as other security tokens is not limited to securing SSH access. The following two websites provide an overview of other possible areas of use:

- *https://github.com/drduh/YubiKey-Guide*
- *https://wiki.archlinux.org/title/YubiKey*

23.6 Additional Tools

For proficient SSH users, there are various additional tools to make the use of SSH more efficient or secure, and I want to briefly introduce you to some of these tools in this section.

23.6.1 Cluster SSH

Cluster SSH (CSSH) is a Perl script that allows you to initiate multiple SSH connections. A separate Xterm window is opened for each connection, and the text color varies depending on the server. The color is not adjustable, but always remains the same for a server. If you use CSSH for a while, you'll be able to tell which server you are working on just by looking at the colors.

But the real highlight of CSSH is that you can type commands not only directly in the terminal windows, but also in a text input box of the tiny CSSH window. This input then is multiplied to all open windows. Thus, if you want to perform an update on three Ubuntu or Debian servers, you must first log in to all three servers using CSSH and then enter "apt dist-upgrade" once in the CSSH control window. This means that the command is executed on all three servers.

Cluster SSH can be installed on many distributions as the `clusterssh` package. If the package is missing in your distribution, you can find Cluster SSH for download at *https://github.com/duncs/clusterssh*.

Unfortunately, Cluster SSH is not compatible with Wayland.

23.6.2 Parallel SSH

The Parallel SSH Python script collection (package name `pssh`) has certain similarities to CSSH. The `pssh` command contained in Parallel SSH is also suitable for executing a command on many servers accessible via SSH—in parallel, that is. The crucial difference from CSSH is that Parallel SSH is not intended for interactive use, but for automated script processing. For this purpose, you first need to save the host names or IP addresses in a text file and then run the required commands on all hosts via Parallel SSH:

```
user$ pssh -h hosts.txt apt-get update
```

If you want to save the output of commands, you can specify an output directory with `-o`. There, `pssh` will save a file with the corresponding name for each host:

```
user$ mkdir allfstabs
user$ pssh -h hosts.txt -o allfstabs cat /etc/fstab
```

Administrative work with `pssh` requires a `root` login on the servers. On Ubuntu, this does not succeed by default. However, you can set a `root` password by using `passwd` and then copy your own key using `ssh-copy-id`. `pssh` requires that authentication with keys is possible or that the background program `ssh-agent` takes care of authentication.

The `pssh` package also provides the `pscp` and `pnuke` commands. You can distribute a file across many computers using `pscp`. The target directory must be specified as an absolute value. `pnuke` terminates a process on all hosts via `kill -9`. You can find examples of using `pssh` at *https://www.geeksforgeeks.org/pssh-linux-comman/*.

23.6.3 Mosh

It's almost impossible to use SSH during a train ride: constant disconnections force you to log in again and again. The solution to this problem is Mosh (*Mobile Shell*), an SSH alternative specifically designed for unstable network connections.

Before you can use `mosh`, you must install the package of the same name on both the server and the client. Then you need to establish the connection in the same way as with `ssh`:

```
localhost$ mosh user@exernalhost
```

`mosh` uses SSH to establish the connection and then starts the `mosh-server` program on the external host with the privileges of the user to whom you have established the connection. Thus, unlike `sshd`, `mosh-server` does not

run as a service all the time, but only when needed. Further communication then takes place via the UDP protocol (not TCP) and via a port between 60000 and 61000. These ports must therefore not be blocked by a firewall.

You can read more details about `mosh` on the project website at *https://mosh.org*.

23.6.4 screen

The `screen` command from the package of the same name enables you to run multiple commands simultaneously in a text console or terminal session and to switch between them. As long as you work locally on a desktop system, the benefit of `screen` is manageable; in this case, it is much easier to open multiple tabs in the terminal window.

`screen` unfolds its full potential in combination with SSH for two reasons: First, you can now switch between *multiple* interactive commands in *one* SSH connection. Second, and this is probably even more important, all commands continue to run if the SSH connection drops (e.g., during a train ride).

Usage First, you want to establish an SSH connection as usual:

```
user$ ssh someuser@remote-host
```

Instead of running commands directly now, you should start `screen` first. Using Ctrl+A, D you can now uncouple a command that's running via `screen` so that it continues to run in the background. In the foreground, start `screen` again and then another command—here, a `for` loop:

```
someuser$ screen                (first screen)
someuser$ ping google.com
someuser$ <Ctrl>+<A>, <D>       (ping contiues to run in the background)
[detached from 3545.pts-0.host1]

someuser$ screen                (second screen)
someuser$ for i in $(seq 1 1000) ; do echo $i ; sleep 1 ; done
someuser$ <Ctrl>+<A>, <D>       (for also continues to run in the background)
[detached from 4264.pts-0.host1]
```

You can obtain an overview of all active screens by using `screen -ls`. Unfortunately, only process numbers are displayed. To reactivate the screen for this process, you must use `screen -r <pid>`:

```
someuser$ screen -ls            (overview of all screens)
There are screens on:
  4264.pts-0.host1      (28.07.2023 13:16:12)   (Detached)
```

```
  3545.pts-0.host1      (28.07.2023 13:13:44)   (Detached)
  2 Sockets in /run/screen/S-kofler.
someuser$ screen -r 3545      (back to the ping screen)
```

If an SSH connection is lost, all processes started via `screen` will continue to run. You can log in again via SSH, determine the currently running commands using `screen -ls`, and switch to their screens using `screen -r <pid>`. Bingo! If you no longer need a particular screen, you should first exit the program running there and then press Ctrl+D.

More application options

`screen` has a lot of other functions. Among other things, you can name your screens, open multiple "windows" within a screen, switch between them, and more. You can find various instructions and (video) tutorials on the internet. The following two pages are a good starting point:

- *https://www.baeldung.com/linux/screen-command*
- *https://www.gnu.org/software/screen/manual/screen.html*

Chapter 24
Apache

This chapter describes how you can set up your own web server. The chapter focuses on the Apache program and its basic configuration, including the use of free HTTPS keys from Let's Encrypt. I also cover some popular extensions, including the PHP programming language and the GoAccess program for creating access statistics. The chapter ends with a short presentation of the Apache alternative, NGINX.

24.1 Apache

Apache versus NGINX

In the open-source world, two web servers are currently in a neck-and-neck race: Apache and NGINX. I focus in this chapter on the Apache program, which is easier to configure and remains the first choice, especially for beginners. In contrast, NGINX (see Section 24.8) is characterized by leaner code and stronger performance for static content.

Installation

A typical Apache installation consists of numerous related packages: the server itself, various libraries, plugins, programming languages, and so forth. To make the installation easier for you, some distributions allow you to select a whole group of packages for installation. In some cases, the most important MySQL and PHP packages are installed in addition to Apache:

```
root# tasksel install web-server                (Debian)
root# dnf groupinstall 'Web-Server'              (Fedora/RHEL)
root# zypper install -t pattern lamp_server      (SUSE)
user$ sudo apt install tasksel                   (Ubuntu)
user$ sudo tasksel install web-server
```

Start/stop

Apache is a background process. On Debian and Ubuntu, Apache is automatically enabled during installation. In other distributions, you have to take care of that yourself:

```
root# systemctl enable --now httpd               (Fedora, RHEL)
root# systemctl enable --now apache2             (SUSE)
```

Firewall

On Fedora, RHEL, and (open)SUSE, the default active firewall blocks access to the web server from the outside. So, for now, you can only try Apache directly on the machine where the web server is running (*http://localhost*). To ensure that the web server can also be reached from outside, you must

define exception rules in the firewall for the HTTP and HTTPS protocols—that is, for port numbers 80 and 443.

On SUSE, it's best to use YaST for configuring the firewall. On Fedora/RHEL, you can proceed as follows: You first determine which firewall zone applies to the network interface to the internet (often `public`), and then enable exception rules for that zone:

```
root# firewall-cmd --get-active-zones   (determine active zone)
public
  interfaces: enp1s0
root# firewall-cmd --permanent --zone=public --add-service=http
root# firewall-cmd --permanent --zone=public --add-service=https
root# firewall-cmd --reload
```

Alternative methods for firewall configuration and a lot of background information on this topic follow in Chapter 29.

Name and account

The service name for the init system varies depending on the distribution. For security reasons, the web server, like most other network daemons, is not executed under the `root` account, but under a different account. The easiest way to determine its name is to use `ps axu`. Table 24.1 summarizes under which name Apache is known to the init system, under which account the program is executed, and where the HTML files are located by default.

Distribution	Process Name	Account	DocumentRoot
Debian, Ubuntu	`apache2`	`www-data`	`/var/www/html`
Fedora, RHEL	`httpd`	`apache`	`/var/www/html`
SUSE	`httpd-<thread-method>`	`wwwrun`	`/srv/www/htdocs`

Table 24.1 Program Name, Account, and DocumentRoot Directory of Apache

Test

To test if everything works, you should start a web browser on the local computer and enter "http://localhost/" or "http://servername/" as the address. You should then see a test page of the web server (see Figure 24.1).

Custom HTML pages

To ensure that the home page of your own website displays instead of the test page, you must save your HTML files in the Apache document directory. This directory is also dependent on the distribution (`DocumentRoot` keyword in the configuration files; see Table 24.1). Your HTML files must be readable for the Apache web server account!

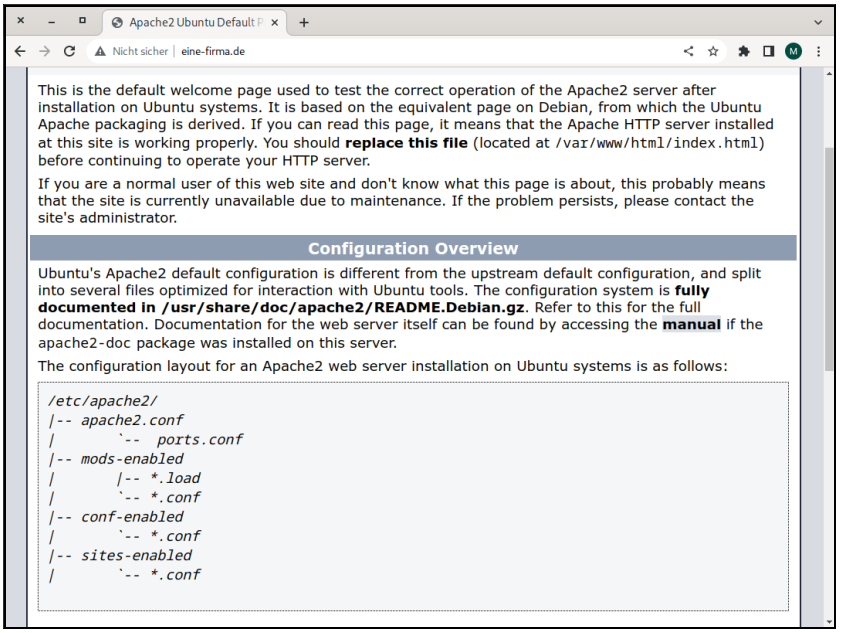


Figure 24.1 Apache Test Page of Ubuntu Server

SELinux

If you work on Fedora or RHEL, you must also make sure that all HTML files have the SELinux attribute `httpd_sys_content_t`. For files inside `/var/www/html`, the easiest way to achieve this is to use the following command:

```
root# restorecon -R -v /var/www/html/*
```

However, if you place your HTML files in a different directory, the following command is required:

```
root# chcon -R system_u:object_r:httpd_sys_content_t:s0 /my-web-directory
```

In both cases, SELinux grants Apache read-only permissions to the files, but there are numerous web applications that require that files may also be modified and added within the installation directory. (WordPress is a well-known program for which these rules apply.) For such programs to work, you must set the SELinux context `httpd_sys_content_rw_t` in the relevant directory:

```
root# chcon -R system_u:object_r:httpd_sys_content_rw_t:s0 dir
```

Note that different attributes are provided for CGI, Webalizer, and configuration files. You can read details about the SELinux module for Apache

by using `man httdp_selinux` if you install the `selinux-policy-doc` package beforehand. Of course, the site can also be found on the internet at *https://linux.die.net/man/8/httpd_selinux*.

24.1.1 Configuration

There is not enough space in this book to go into detail about all configuration options and variants. However, I want to give you at least an overview of where the configuration files are located depending on the distribution and how very elementary settings can be carried out.

Apache used to be configured using a single `httpd.conf` file, the exact location of which depended on the distribution. This configuration file became more and more confusing over time. For this reason, most distributions have moved to distributing the settings to various files that are read by `include` statements from various directories (see Table 24.2 to Table 24.4). This makes each individual file clearer and enables automated maintenance—for example, activating or deactivating plugins through commands or scripts. If you are looking for a specific keyword in the configuration files, it is best to proceed as follows:

```
user$ cd /etc/httpd (or) cd /etc/apache2
user$ grep -i -r keyword
```

Files	Contents
/etc/apache2/apache2.conf	Starting point
/etc/apache2/httpd.conf	User-specific configuration
/etc/apache2/ports.conf	Monitored ports, usually port 80
*/etc/apache2/conf.d/**	More configuration files
/etc/apache2/mods-available/	Available extension modules
/etc/apache2/mods-enabled/.conf*	Links to active extension modules
/etc/apache2/conf-available/	Available configuration files
/etc/apache2/conf-enabled/.conf*	Links to active configuration files
/etc/apache2/sites-available/	Available websites (virtual hosts)
/etc/apache2/sites-enabled/.conf*	Links to active websites
/etc/apache2/envvars	Environment variables for the init script

Table 24.2 Apache Configuration on Debian and Ubuntu

Files	Contents
/etc/httpd/conf/httpd.conf	Starting point
/etc/httpd/conf/magic	MIME configuration (for `mod_mime`)
/etc/httpd/conf.d/.conf*	Other configuration files

Table 24.3 Apache Configuration on Fedora and RHEL

Files	Contents
/etc/apache2/httpd.conf	Starting point
/etc/apache2/.conf*	Global configuration files
/etc/apache2/conf.d/.conf*	Other configuration files
/etc/apache2/sysconfig.d/.conf*	System configuration files
/etc/apache2/vhosts.d/.conf*	Websites (virtual hosts)
/etc/sysconfig/apache2	Basic settings

Table 24.4 Apache Configuration on SUSE

Debian/Ubuntu

On Debian/Ubuntu, the `mods-available` directory contains a collection of `*.load` and `*.conf` files for various Apache modules. To enable additional modules, you need to set up links to these files in `mods-enabled`. The Debian-specific `a2enmod` and `a2dismod` commands help you to manage the links. You can use `a2ensite` and `a2dissite` to enable or disable virtual hosts. By default, `sites-available` contains only the `000-default.conf` and `default-ssl.conf` files: various basic settings for the `/var/www` directory can be found there. The mechanism works in the same way as with the modules: The `sites-available` directory contains the configuration files for all hosts, while `sites-enabled` contains the corresponding links.

The same mechanism takes care of other configuration files. The files are located in the `conf-available` directory. The `a2enconf` and `a2disconf` commands are used to create and remove links to the `conf-enabled` directory. On older Ubuntu versions as well as Debian, such configuration files are stored in `conf.d` and are managed manually without commands.

SUSE

On SUSE, all `*.conf` files in the `sysconf.d` directory are recreated by the init system at each Apache startup. It is therefore pointless to make changes to these files. Rather, you must change the variables in `/etc/sysconfig/apache2`. This file also defines which modules are loaded when Apache is started (`APACHE_MODULES` variable). If you want to add your own file to the

SUSE configuration files, you must specify its filename in the APACHE_CONF_INCLUDE_FILES variable.

Testing the configuration

After making changes to the syntax, you can use httpd -t, httpd2 -t, or apache2 -t to test whether the configuration is free of syntax errors. On Debian and Ubuntu, you have to load some environment variables from envvars prior to that:

```
root# . /etc/apache2/envvars
root# apache2 -t
Syntax OK
```

Then you want to prompt Apache to reload the configuration files:

```
root# systemctl reload apache2   (Debian, SUSE, Ubuntu)
root# systemctl reload httpd     (Fedora, RHEL)
```

After making changes to the SSL configuration, systemctl reload is not sufficient. You need to run systemctl restart.

ServerName

The Apache web server usually works right away. However, depending on your network configuration, you may often need to change or add a line to the configuration files. ServerName should contain the name of your computer. If this setting does not take effect, you must also use the UseCanonicalName Off setting:

```
# in   /etc/apache2/apache2.conf    (Debian/Ubuntu)
# or /etc/httpd/conf/httpd.conf   (Fedora/RHEL)
# enter the name of your computer here:
ServerName example.com
```

For SUSE, you need to set the host name in /etc/sysconfig/apache2 with the APACHE_SERVERNAME variable.

24.1.2 Default Character Set

All common Linux distributions automatically use the Unicode UTF-8 character set. So if you use a text editor to create a text file that contains German letters like ä, ö, ü, or ß or French letters like ç or é, they will be saved in UTF-8 encoding.

Apache generally doesn't care about the coding of the HTML files. The program simply transfers the files byte by byte to the web browser that requested the page. However, Apache also sends a so-called header, which, among other things, contains information about the character set in which the page is encoded. The web browser analyzes this information and uses the specified character set to display the page.

Setting the character set

The crucial point is that Apache specifies the correct character set: if this goes wrong, the user will see some strange character combinations in the web browser instead of ä or ü. For this reason, Apache provides various character set configuration options:

- **`AddDefaultCharset off`**
  This setting causes Apache to analyze the `<meta>` tag in the HTML file to be transferred and send the character set specified there to the browser. If the HTML file starts as follows, the Unicode UTF-8 character set will be applied:

```
<!DOCTYPE html>
<html>
 <head>
 <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
 ...
```

- **`AddDefaultCharset character set`**
  Apache transfers the character set specified here to the browser for all pages. The setting applies to both HTML and PHP files. The `<meta>` tag in the HTML code will be ignored.
- **`AddCharset character set identifier`**
  This sets a character set for files with a specific identifier. `AddCharset utf-8 .utf8` thus causes Unicode UTF-8 to be sent to the browser as the character set for all files whose name ends in `.utf8`. `AddCharset` requires the Apache module, `mod_mime`.

Debian, Ubuntu

Of course, a different default configuration will apply for different distributions. For the global default value of the character set, the `/etc/apache2/conf-enabled/charset.conf` configuration file is provided on Debian and Ubuntu. Usually, this file is empty; that is, `AddDefaultCharset off` applies.

You can also use `AddDefaultCharset` and `AddCharset` in virtual host configuration files (`sites-available` directory) and in `.htaccess` files if you want a host-specific or directory-specific configuration. Note, however, that the character set settings in `.htaccess` are only taken into account if `AllowOverride All` or `FileInfo` applies to the web directory.

Fedora, RHEL

For Fedora and RHEL, `AddDefaultCharset UTF-8` applies. The setting is located in `/etc/httpd/conf/httpd.conf`. In the same file, `AllowOverride None` is also set for the `/var/www/html` directory.

SUSE

SUSE lacks an explicit character set setting in the configuration files. This means that `AddDefaultCharset off` applies; that is, the `<meta>` information in

the HTML files is decisive for the correct character set recognition. A suitable place to set `AddDefaultCharset` is the `/etc/apache2/mod_mime-defaults.conf` file. Also on SUSE, `AllowOverride None` applies to the `/srv/www/htdocs` directory. You can change the setting in `/etc/apache2/default-server.conf`.

24.1.3 Logrotate

Apache's logging files are among the fastest growing files on many servers. That's why you need to take care of renaming, compressing, and finally deleting the logging files on a regular basis. This is exactly the task performed by the `logrotate` program (see Chapter 14, Section 14.12.4), which is usually installed by default on Linux servers.

The program is started once a day by Cron or by the systemd timer. In the default configuration, it processes the `/var/log/httpd/*.log` (Fedora/RHEL) or `/var/log/apache2/*.log` (Debian/Ubuntu) Apache logging files once a week, renames them to `name.nn`, and compresses them. The compressed files are archived for 52 weeks and then deleted.

If you define your own logging directories when configuring virtual hosts, you must modify the first line in the `/etc/logrotate.d/apache2` configuration file to specify the locations of the additional logging files.

24.2 Encrypted Connections (HTTPS)

In the default configuration, Apache either uses only the unencrypted HTTP protocol, or it performs the encryption required for HTTPS using a self-signed certificate. Both variants are rightly considered insecure by all major web browsers. If the web browser displays your pages at all, it will show a crossed-out lock or other security warnings in the address bar.

HTTPS

The solution is to use HTTPS in combination with a certificate. HTTPS combines the Hypertext Transfer Protocol (HTTP) and Secure Sockets Layer (SSL) protocols, adding encryption capabilities to HTTP.

You can obtain a suitable certificate for Apache free of charge from organizations such as Let's Encrypt or ZeroSSL. Alternatively, you can purchase a certificate from an established certificate provider. This is inconvenient and expensive, but it gives your customers the assurance that not only the data exchange is secure, but that the website is really operated by you (by your company, by your organization) and not by someone else.

This section provides an overview of important HTTPS basics and related Apache settings. Then, in Section 24.3, you will find specific instructions on

how to set up a Let's Encrypt certificate on your server. If you're in a rush and want to deal with the basics another time, you can skip straight to the next section.

24.2.1 Certificates

Before you can start the configuration work, you need a server certificate. At this point, I have to elaborate a little.

Basic principles

The encryption of the data is based on asymmetric encryption methods. The basic idea is that there is a key pair consisting of a private (secret) key and a public key. The public key is only suitable for *encrypting* data. The private key is only required for *decrypting* data. I won't go into the details of these methods here as they have already been described and explained countless times, including on Wikipedia.

When a connection is established between the client (i.e., a web browser) and the server (Apache), a shared key is first negotiated on the basis of a random number from the client and the public key from the server (handshake method). This *session key* is then used to encrypt all further communication. The web browser is thus able to encrypt the data to be sent in such a way that only the web server can analyze it by means of its private key.

However, data exchange that is virtually tap-proof according to current knowledge is only *one* aspect of improving security. Another aspect is that a user must be certain that they are communicating with the right partner. What good is it if the data for online banking is transmitted in a tap-proof manner, but instead of going to the bank it ends up directly in the hands of a fraudster?

For this reason, certificates can also contain data about the website and some kind of signature of a certification body. Their task is to verify the identity of the certificate applicant by means of a passport copy, a trade license, or similar. Unfortunately, this control process makes such certificates relatively expensive.

It depends on the trustworthiness of the authentication authority how trustworthy a certificate is. Well-known web browsers such as Firefox or Google Chrome only accept certificates issued by established authentication organizations (such as Verisign, Thawte, or Let's Encrypt). For other certificates, obvious warnings are displayed.

File identifiers

A key or certificate is ultimately a simple text file that contains some metadata as well as the actual key as a Base64-encoded string. Table 24.5 summarizes the common file identifiers for key and certificate files. Here, `pem`

stands for *privacy enhanced mail*. This is a method for encrypted emails that has not caught on. However, the PEM format defined there is still common today. Usually, `*.crt` files contain `pem` files as well.

Identifier	Meaning
`*.key`	Private key file
`*.csr`	Unsigned certificate (Certificate Signing Request)
`*.pem`	Container file for a signed certificate or certificate chain
`*.crt`	`*.pem` file with a different identifier (Windows)

Table 24.5 File Identifiers for Key and Certificate Files

24.2.2 Using Self-Signed Certificates

You can generate certificates yourself using the `openssl` command. This kind of certificate is not considered trustworthy, but it provides the opportunity to become familiar with certificates and to get to know the `openssl` command. If you are not interested in this, you can skip this section. The knowledge imparted here is not absolutely necessary for the use of ready-made certificates from Let's Encrypt and the like.

In many distributions, self-signed certificates are even available by default. On Debain and Ubuntu, they are referred to as *snake-oil certificates* (`/etc/ssl/certs/ssl-cert-snakeoil.pem`). Fedora and RHEL instead have a self-signed certificate for the hostname of the machine (`/etc/pki/tls/certs/localhost.crt`).

openssl

First, you need to install the `openssl` package. It contains the command of the same name for generating, managing, and manipulating keys and certificates:

```
root# apt/dnf/zypper install openssl
```

First the key...

The following command creates a file with a private 2048-bit RSA key. You will need this key twice, once to generate a certificate signing request and then again to sign it:

```
root# openssl genpkey -algorithm RSA -pkeyopt rsa_keygen_bits:2048 \
        -out server.key
```

Make sure that only `root` can read this file! If this file falls into foreign hands, your server certificate will be worthless and you will have to revoke it:

```
root# chmod 400 server.key
```

Encrypted Keys

Keys are valuable, which is why they are usually encrypted themselves by a *passphrase*, a particularly long password. This passphrase is also referred to as an *encrypted key*. If you want to use such a key, you need to add the `-aes-256-cbc` option to the `openssl` command. You must then enter this password each time you use the key for the other commands.

This is also the case for Apache: the web server needs the key to read the certificate. That's why Apache now asks for the key's password at every (re)start. This makes an automated restart, as during an update, impossible.

To avoid this problem, an unencrypted copy of the key is created for Apache (`openssl rsa -in server.key -out server-unencrypted.key`). Apache is then configured to use the insecure key file instead of the encrypted key, which must therefore be stored in the server's file system. And now, at the latest, it becomes clear that an encrypted key in our case is just an unnecessary extra effort that does not increase security. Of course, this does not mean that encrypted keys are generally not recommended—quite the opposite!

... then the certificate...

It's a bit more work to create the certificate. Strictly speaking, the following command does not generate the final certificate, but a *certificate signing request*. The `server.key` key is also included in the certificate (`-key` option).

When running the command, you must specify in which country and place you live, what your name is, and so on. The crucial question is the *common name*: in this context, it is not your name that is required, but the exact name of your website as it is used for encrypted connections.

Some websites use a separate subdomain for encrypted connections (e.g., `banking.<mybank>.com`); others do not (such as `<mybank>.com`). However, the certificate is only valid for a certain spelling. So, for example, you cannot use a certificate for `www.a-company.com` also for `a-company.com` (or vice versa)! Make sure that you leave the challenge password blank; that is, answer the corresponding question by entering a dot:

```
root# openssl req -new -sha256 -key server.key -out server.csr
Enter pass phrase for server.key: ******************
You are about to be asked to enter information that will be
incorporated into your certificate request.

What you are about to enter is what is called a Distinguished
Name or a DN. There are quite a few fields but you can leave
some blank. For some fields there will be a default value,
```

```
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [US]:            EN
State or Province Name (full name) [...]:     .
Locality Name (eg, city) []:                  New York
Organization Name (eg, company) [Sample Inc.]: John Doe
Organizational Unit Name (eg, section) []:    .
Common Name (eg server FQDN or YOUR name) []: www.a-company.com
Email Address []:                             webmaster@a-company.com

Please enter the following 'extra' attributes
to be sent with your certificate request
A challenge password []:                      .
An optional company name []:                  .
```

... and finally the signature

The next command enables you to sign your certificate yourself. With a "real" certificate, this process is carried out by the authentication authority—of course, after checking the documents you have submitted. The key of the authentication authority is then used for the signature.

By default, the ready certificate is valid only for 30 days. The `-days 1900` option extends the validity period to approximately five years:

```
root# openssl x509 -req -days 1900 -in server.csr  \
        -signkey server.key -sha256 -out server.pem
Signature ok
subject=/C=US/L=New York/O=John Doe/CN=www.a-company.com/
  emailAddress=webmaster@a-company.com
Getting Private key
```

Unable to Write Random State

Sometimes the *unable to write random state* error occurs when you execute the above command. The reason for this is mostly that `openssl` does not find the home directory—often because you have previously run `sudo -s`. The solution to this is to run `export PATH=/root`.

Checking

Physically, the generated keys and certificates are relatively small text files. To display the data contained in a certificate in plain text, you can use the `openssl x509 -text` command. The following issues are shortened due to space constraints:

```
root# ls
  server.key  (unencrypted private key)
  server.csr  (certificate without signature)
```

server.crt (certificate with signature)

```
root# cat server.pem
-----BEGIN CERTIFICATE-----
MIICWTCCAcICCQCL6ExhrQiELDANBgkqhkiG9w0BAQUFADBxMQswCQYDVQQGEwJB ...
-----END CERTIFICATE-----

root# openssl x509 -text -in server.pem
Certificate:
    Data:
        Version: 1 (0x0)
        Serial Number: 12669601459972319941 (0xafd37766c36baac5)
    Signature Algorithm: sha256WithRSAEncryption
        Issuer: C=US, L=New York, O=John Doe,
                CN=www.a-company.com/emailAddress=
webmaster@a-company.com
        Validity
            Not Before: Sep 28 14:48:03 2023 GMT
            Not After : Dec 10 14:48:03 2028 GMT
        Subject: C=US, L=New York, O=John Doe,
                 CN=www.a-company.com/emailAddress=
webmaster@a-company.com
        Subject Public Key Info:
            Public Key Algorithm: rsaEncryption
                Public-Key: (2048 bit)
                Modulus:
                    00:be:37:21:23:b6:13:e4:92:74:36:9f:de:4e:9a:
    Signature Algorithm: sha256WithRSAEncryption
         43:b8:92:82:0d:f6:e2:ef:ff:07:eb:3a:1f:da:3d:d9:ba:53:
         d7:1f:4a:49:ec:5a:c1:fb:0f:95:e8:94:89:ab:7b:05:95:62:
```

24.2.3 Apache Configuration for HTTPS Operation

mod_ssl

The Apache functions required for the HTTPS protocol are located in the mod_ssl module. On Debian or Ubuntu, this module is installed by default and merely needs to be activated:

```
root# a2enmod ssl
root# a2ensite default-ssl
root# systemctl restart apache2
```

On Fedora or RHEL, you must first install the SSL module:

```
root# dnf install mod_ssl
root# systemctl restart httpd
```

SSL configuration

Apache needs to read the key and certificate files, so the obvious thing to do is to copy the two files to the Apache configuration directory:

```
root# cp server.pem server.crt /etc/apache2   (Debian, SUSE, Ubuntu)
root# cp server.pem server.crt /etc/httpd     (Fedora, RHEL)
```

Next, you need to change the default SSL configuration. `SSLEngine on` activates the encryption functions. `SSLxxxFile` specifies where the files with the certificate and the private key are located. The distinction between HTTP and HTTPS is made by the port number 443 set in the `VirtualHost` line:

```
# Change SSLCertificateFile and SSLCertificateKeyFile,
# e.g. in /etc/httpd/conf.d/ssl.conf (Fedora, RHEL)
# or in /etc/apache2/sites-available/default-
ssl.conf (Debian, Ubuntu)
<VirtualHost _default_:443>
    ServerName             www.a-company.com
    DocumentRoot           /var/www/html/
    SSLEngine              on
    SSLCertificateFile     /etc/apache2/server.pem
    SSLCertificateKeyFile  /etc/apache2/server.key
</VirtualHost>
```

To enable the HTTPS site, you must prompt Apache to reread the configuration. If you've done the HTTPS configuration on Debian/Ubuntu in a separate file in `/etc/apache2/sites-available`, you need to enable this file:

```
root# systemctl restart httpd        (Fedora/RHEL)
root# a2ensite ssl-a-company         (Debian/Ubuntu, part 1)
root# systemctl restart apache2      (Debian/Ubuntu, part 2)
```

If you switch from an operation without SSL to an operation with SSL for the first time, it is not sufficient to reload the configuration files. For the SSL module to be loaded, you must specify `restart` instead of `reload`!

Safety warning

When a web browser encounters a self-signed certificate or a snake-oil certificate, the browser displays a security warning, as in Figure 24.2.

The warning does not refer to the encryption not being secure: it is! Rather, the web browser warns you because the identity of the entity that signed the certificate is unknown. Technically unsophisticated users will therefore consider the website insecure. However, if you only want to be able to use phpMyAdmin securely to administrate the MySQL installation on the server, for example, then this certificate is completely sufficient!

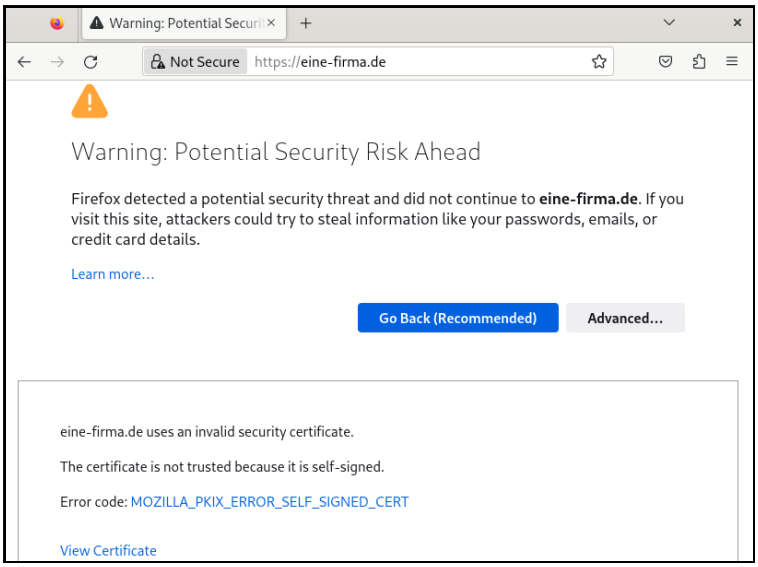


Figure 24.2 Warning about Self-Signed Certificate

24.2.4 Snake-Oil Certificates

Debian and Ubuntu

For Debian and Ubuntu, a snake-oil key and a snake-oil certificate valid for 10 years are automatically generated when Apache is installed:

```
/etc/ssl/certs/ssl-cert-snakeoil.pem      (certificate)
/etc/ssl/private/ssl-cert-snakeoil.key    (key)
```

As you know, the term *snake oil* is used to describe supposed miracle cures or panaceas. The certificate is generated to allow web and mail servers to be used immediately in encrypted mode without the inconvenience of generating your own certificates. Accordingly, the Apache configuration file `default-ssl.conf` contains the following lines:

```
# Datei /etc/apache2/sites-available/default-ssl.conf
...
SSLCertificateFile    /etc/ssl/certs/ssl-cert-snakeoil.pem
SSLCertificateKeyFile /etc/ssl/private/ssl-cert-snakeoil.key
```

The same restrictions apply to the snake-oil certificate as to certificates that you have generated and signed yourself using `openssl`—hence the name. The snake-oil certificate takes into account the host name (`/etc/hostname`) that was valid when it was created. If you need a new certificate after changing a host name, you can call the following command:

```
root# make-ssl-cert generate-default-snakeoil --force-overwrite
```

`make-ssl-cert` is a relatively simple script that uses the `openssl` command described earlier.

Fedora and RHEL

Distributions from the Red Hat family have similar, automatically created and self-signed default certificates:

```
# file /etc/httpd/conf.d/ssl.conf
...
SSLCertificateFile    /etc/pki/tls/certs/localhost.crt
SSLCertificateKeyFile /etc/pki/tls/private/localhost.key
```

The certificates are valid for one year for the host name that was set during the installation of the `httpd` package.

24.3 Let's Encrypt

Since 2016, the *Let's Encrypt* organization (*https://letsencrypt.org*) has been offering free certificates. These certificates are very widely used. By mid-2023, there were over 300 million active certificates from Let's Encrypt in use. At the same time, there are now other providers for free certificates, such as ZeroSSL. So the obvious question is, "Why spend any more money on third-party certificates?" Let's Encrypt comes with two major limitations: For one thing, the certificates are only valid for 90 days. To get around this disadvantage, the system must be set up in such a way that the certificates are automatically updated on a regular basis. (This is not difficult.) On the other hand, the certificates from Let's Encrypt are pure domain certificates. They allow you to check that a website actually comes from the *a-company.com* domain, for example, and not from another server that only pretends to do so.

However, Let's Encrypt does not perform any checks to see who owns this domain (*organization validation*), let alone who this person or organization is (*extended validation* by checking a passport copy, employer ID, etc.). In other words, impostors can also use free certificates from Let's Encrypt.

Certificates from Let's Encrypt are comparable to the cheapest and at the same time most widespread certificate variant of the established certification authorities. The certificates are sufficient to encrypt your own blog or similar pages. However, companies or organizations that implement a store on their websites, offer online banking services, manage health or insurance data, and so on are still dependent on higher-quality and thus unfortunately also quite expensive certificates from other providers.

Certificate management software

In the early months, Let's Encrypt itself offered software for certificate management, including the now obsolete `letsencrypt` command. Now the

Electronic Frontier Foundation (EFF) takes care of the development and maintenance of the software. The official Let's Encrypt client is called `certbot` (*https://certbot.eff.org*). `certbot` was a kind of factual standard in the past due to a lack of alternatives. In recent years, however, the program has repeatedly caused massive problems, partly due to its complicated installation.

For this reason, I recommend the `acme.sh` shell script. Compared to `certbot`, this script is much more straightforward to install and easier to use. However, it has fewer special functions. Besides `certbot` and `acme.sh`, there are a number of other tools that can assist you with setting up certificates, including `getssl`. A list of other clients, including download links, can be found at *https://letsencrypt.org/docs/client-options*.

24.3.1 Installing acme.sh

`acme.sh` is not available as a ready-made package. Nevertheless, the installation succeeds without any problem by using the commands ahead. You need to download a setup script from the project website and run it with `root` privileges. (Whenever you do something like this, you should take a look at the script first! Maybe some bad guy took over or manipulated the download page?) Then you pass an email address to the setup program. Let's Encrypt sends warning emails to this email address if the automatic certificate renewal should not work:

```
user$ sudo apt/dnf/zypper install curl socat tar
user$ curl https://get.acme.sh -o acme-setup
user$ less acme-setup      (briefly check the script code)
user$ sudo sh acme-setup email=admin@a-company.com
...

Installed to /root/.acme.sh/acme.sh
Installing alias to '/root/.bashrc'
Installing cron job
OK, Close and reopen your terminal to start using acme.sh
```

After the installation, you must log out and log in again so that the new alias for calling acme.sh works.

24.3.2 Applying acme.sh

You need to run the two most important `acme.sh` commands with the `--issue` and `--install-cert` options.

Issuing a certificate

`acme.sh --issue` requests a common certificate from Let's Encrypt for all domain names specified with the `-d` option. The `-w` option specifies the web root directory. `acme.sh` temporarily stores a small file there, which the Let's Encrypt server then tries to read via a web call. This ensures that you really have access to the specified domains. (Without such checks, anyone could generate a certificate for *deutsche-bank.de* at Let's Encrypt, for example. Of course, this must be prevented.) `acme.sh` issues certificates from the ZeroSSL organization by default. If you prefer Let's Encrypt certificates, you must specify the `--server letsencrypt` option.

The following command generates a certificate for the web presence of `a-company.com`, also taking into account the alternative address `www.a-company.com`. Before calling the command, you should make sure, such as by using `ping`, that all the specified addresses actually reference your server—that is, that the DNS configuration is correct! As this check uses HTTP or HTTPS protocols, you must enable ports 80 and 443 before running `acme.sh` if this has not already been done:

```
root# acme.sh --issue  --server letsencrypt  -d a-company.com \
        -d www.a-company.com -w /var/www/html
Using CA: https://acme-v02.api.letsencrypt.org/directory
Creating domain key
The domain key is here: /root/.acme.sh/a-company.com/
a-company.com.key
Multi domain='DNS:a-company.com,DNS:www.a-company.com'
Verifying: a-company.com
Verifying: www.a-company.com
Downloading cert.

Cert success.

Your cert is in                /root/.acme.sh/a-company.com/
a-company.com.cer
Your cert key is in            /root/.acme.sh/a-company.com/
a-company.com.key
The intermediate CA cert is in /root/.acme.sh/a-company.com/ca.cer
The full chain certs is there  /root/.acme.sh/a-company.com/
fullchain.cer
```

Note that you must replace `/var/www/html` with the path where your web server is looking for the files for the specified host. The path varies depending on which distribution you are running and how you have integrated multiple hosts into the Apache configuration, if applicable. The outputs of the command have been reproduced here in highly shortened form for

reasons of space. You can see that the key files are stored by default in a subdirectory of /root/.acme.sh/a-company-com.

Wildcard certificates as a special case

Let's Encrypt and acme.sh also support wildcard certificates that are valid for all conceivable subdomains of a domain. The use of wildcard certificates is only appropriate if all subdomains are served by *one* server, which is rather unusual for large setups.

Issuing a wildcard certificate is more complicated and involves two steps. In the first step, you want to specify the -d option twice, once for the primary domain and once for all subdomains—for example, like this:

```
-d a-company.com -d '*.a-company.com'
```

The acme.sh script then prompts you to manually add two TXT records to your domain as an additional verification measure. Once the DNS changes are effective, you must run acme.sh again in the second step, this time with the --renew option. The required syntax of the second command can be found in the output of the first command.

Unfortunately, the manual verification will also be required for every certificate renewal in the future, which is extremely inconvenient. For many DNS providers, acme.sh supports an API to automate the process. Details are available in the documentation at *https://github.com/acmesh-official/acme.sh/wiki/dnsapi*.

Copying certificates to the target directory

Setting up the certificates is not enough as they are currently located only in the /root/.acme.sh directory, which the web server is not allowed to access. It is not a good idea to copy the certificates manually to another directory now; this process would have to be repeated for each certificate renewal.

Instead, you should complete this task using acme.sh --install-cert. This has the advantage that acme.sh remembers all the information and automatically takes care of copying renewed certificates to the specified location again in the future. The following example provides /etc/acme-letsencrypt as the location for the certificates and copies the certificates from a-company.com there. The command specified with --reloadcmd is executed after copying and every time after a certificate update in the future. On Fedora/RHEL, you must replace apache2 with httpd:

```
root# mkdir /etc/acme-letsencrypt
root# acme.sh --install-cert -d a-company.com \
     --cert-file      /etc/acme-letsencrypt/a-company.com.cert \
     --key-file       /etc/acme-letsencrypt/a-company.com.key \
     --fullchain-file /etc/acme-letsencrypt/a-company.com.fullchain \
     --reloadcmd 'systemctl restart apache2'
```

```
Installing cert to:       /etc/acme-letsencrypt/a-company.com.cert
Installing key to:        /etc/acme-letsencrypt/a-company.com.key
Installing full chain to: /etc/acme-letsencrypt/a-company.com.fullchain
Run reload cmd: systemctl restart apache2
Reload success
```

`acme.sh` does not change the configuration files of your web server! You must therefore add the appropriate statements with the paths to the certificate files themselves. The relevant lines for the preceding example look as follows:

```
# e.g. in /etc/apache2/sites-enabled/default-
ssl.conf (Debian, Ubuntu)
# or  in /etc/apache2/hosts-enabled/a-company.com.conf
# or  in /etc/httpd/conf.d/ssl.conf                  (Fedora, RHEL)
SSLCertificateFile      /etc/acme-letsencrypt/a-company.com.fullchain
SSLCertificateKeyFile   /etc/acme-letsencrypt/a-company.com.key
```

Renewing certificates

During the installation of `acme.sh`, one line was added to `/var/spool/cron/crontabs/root` or `/var/spool/cron/root` depending on the distribution. This makes Cron take care of calling `acme.sh --cron` on a regular basis and thus automatically renewing all certificates set up by using `acme.sh`. Any certificate older than 60 days will be automatically renewed. For the web server to take the changed certificates into account, it is automatically restarted (using the `reload` command you specified along with `acme.sh --install-cert`).

To test the renewal process, you can run `acme.sh --cron --force`. Note, however, that the number of Let's Encrypt transactions per day is limited.

Changing/deleting certificates

If you need a certificate for another subdomain, you can request either a new certificate for this subdomain only or a new joint certificate that combines all previous domains plus the new domain. In the first case, you have two independent certificates that you can use separately, which gives you more flexibility.

Using `acme.sh --remove -d` *name*, you can deactivate the renewal for a domain, which takes place automatically every 60 days. Although the name of the option suggests otherwise, the certificates themselves will not be deleted. However, you can now delete the corresponding `/root/.acme.sh/name` directory yourself.

Redirection from HTTP to HTTPS

It is often preferable for all HTTP page accesses to be automatically redirected to HTTPS. If you pass the `--redirect` option, `certbot` will integrate the corresponding statements right into the Apache configuration files. Of

course, you can also do this yourself. The following lines provide a pattern for this:

```
<VirtualHost _default_:80>
  ...
  RewriteEngine on
  # Redirection from HTTP to HTTPS
  RewriteCond %{SERVER_NAME} =www.a-company.com [OR]
  RewriteCond %{SERVER_NAME} =a-company.com
  RewriteRule ^ https://%{SERVER_NAME}%{REQUEST_URI} [END,NE,R=
permanent]
</VirtualHost>
```

If you also want to redirect from *www.a-company.com* to *a-company.com*, you need to add the following lines:

```
# redirection from www.<hostname> to <hostname>
# source: http://stackoverflow.com/questions/21467329
RewriteCond %{HTTP_HOST} ^(www\.)(.*) [NC]
RewriteRule (.*) https://%2%{REQUEST_URI} [L,R=301]
```

24.3.3 SSL Settings

Additional SSL options

Setting up certificates is not the end of the story. There are countless encryption methods and versions, all of which are summarized under HTTPS. Usually, Apache and the web browser exchange information about which protocols they each support and then use the most secure method that both can handle. Some methods are unsafe from today's perspective and should therefore be blocked. For this reason, it is advisable to use `SSLCipherSuite` and `SSLProtocol` to explicitly specify which procedures your website supports and which ones it rejects.

The *https://ssl-config.mozilla.org* website is a great help for optimal configuration. There you can specify which version of the web server you are running and how important the compatibility with old browser versions is to you. Based on these few pieces of information, the website generates a listing with settings that you only need to transfer to your server configuration (see Figure 24.3). Note that you must include the settings lines in the correct places:

```
# in the <VirtualHost> section:
# enable HTTP Strict Transport Security (HSTS) and HTTP/2
Header always set Strict-Transport-Security  "max-age=63072000"
Protocols h2 http/1.1
...
```

```
# outside the <VirtualHost> section, applies to the entire web server:
# SSL intermediate configuration, see https://ssl-config.mozilla.org
SSLProtocol             all -SSLv3 -TLSv1 -TLSv1.1
SSLCipherSuite          ECDHE-ECDSA-AES128-GCM-SHA256:ECDHE-RSA-AES...
SSLHonorCipherOrder     off
SSLSessionTickets       off
SSLUseStapling          on
SSLStaplingCache        "shmcb:logs/ssl_stapling(32768)"
```

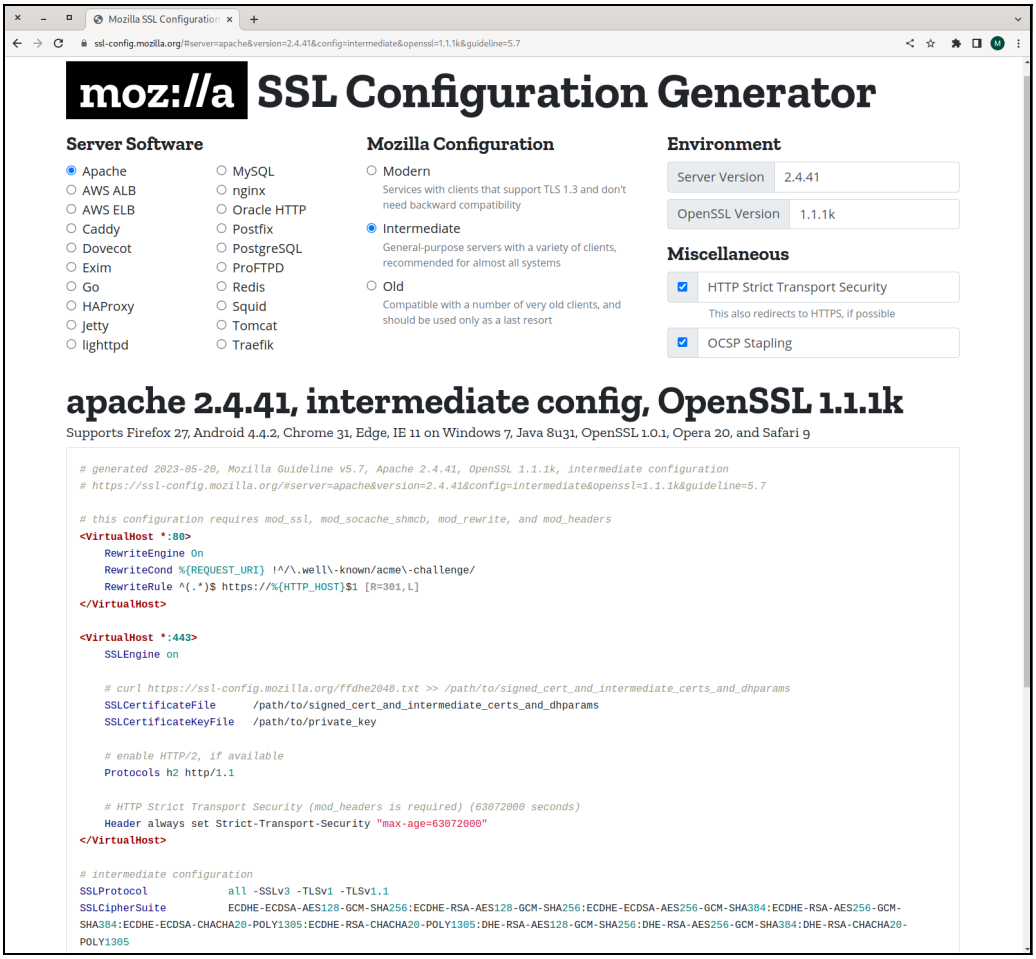


Figure 24.3 SSL Configuration Generator

The configuration described here is based on the assumption that various Apache add-ons are available. These must be explicitly activated depending on the distribution. The following commands apply to Debian and Ubuntu:

```
root# a2enmod headers rewrite socache_shmcb
root# systemctl restart apache2
```

SSL test

To verify that your configuration is working, you want to enter the address of your web server at *https://www.ssllabs.com/ssltest*. A script then checks the security of your website and provides optimization tips (see Figure 24.4).

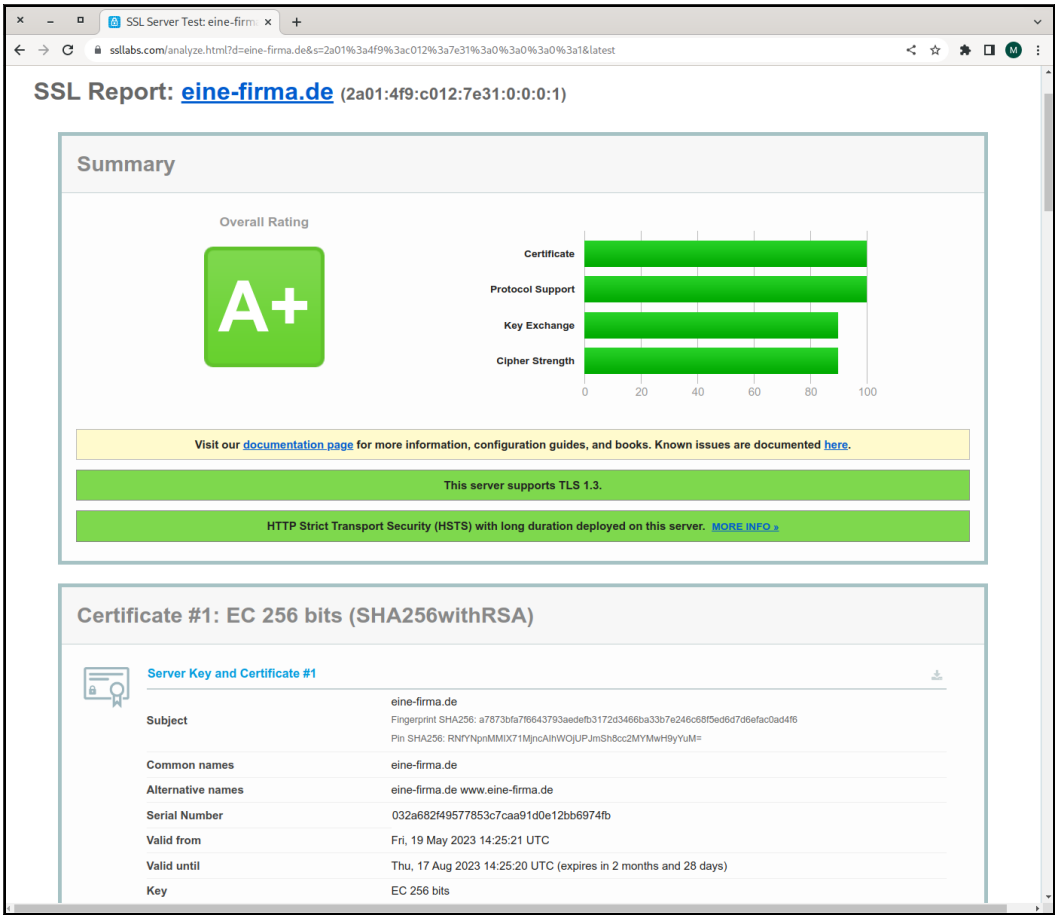


Figure 24.4 Online Verification of HTTPS Configuration

24.4 Setting Up and Securing Web Directories

After the basic configuration of Apache, you will usually set up various web directories that contain those HTML and PHP files that make up your website. For example, if you want to set up WordPress as the CMS for your website, you download the installation files, set up a directory accessible to Apache, and unpack the files there. This section explains what settings you need to make in Apache for the directories where you want to set up WordPress, phpMyAdmin, NextCloud, or any other web application.

The following pages are based on the configuration as you will find it on Ubuntu or Debian. If you use a different distribution, there are small variations in the default configuration. However, the keywords and work techniques presented here apply there as well.

Default Ubuntu configuration

On Ubuntu, Apache is preconfigured to use files from the `/var/www/html` directory for the default website. The required settings are located in the `/etc/apache2/sites-available/000-default.conf` file:

```
# file /etc/apache2/sites-available/000-default.conf (Ubuntu)
<VirtualHost *:80>
  ServerAdmin      webmaster@localhost
  DocumentRoot     /var/www/html
  ErrorLog         ${APACHE_LOG_DIR}/error.log
  CustomLog        ${APACHE_LOG_DIR}/access.log combined
</VirtualHost>
```

A Debian- or Ubuntu-specific feature of the default configuration is that all settings are bundled in a `<VirtualHost>` group. `<VirtualHost>` groups are used to separate settings for several independent hosts (websites; see Section 24.5). However, the host in the `default` file is not linked to either an IP address or a host name, and for this reason applies to all web accesses that cannot be assigned to a specific virtual host.

24.4.1 Host Configuration

The keywords described ahead for configuring a `<VirtualHost>` group are used to specify the details of the host—the origin of the data, the administrator's email address, the location of the logging files, and so on:

- `DocumentRoot` specifies the directory in which the HTML files are located.
- `ServerAdmin` specifies the email address of the virtual host administrator. The address is displayed in case of error messages, for example. Here you should enter an email address that is actually active. A commonly used address is `webmaster@hostname`.
- `ServerSignature` controls whether Apache should add a signature at the end of self-generated documents (error messages, directory listings, etc.). The signature consists of the Apache version and the host name. `ServerSignature=EMail` also adds the administrator's email address.
- `LogLevel` determines to what extent web server problems should be logged. Possible values range from `emerg` (log only critical errors that cause Apache to terminate) to `debug` (log everything, even debugging texts). Usually, the `error` or `warn` settings make much sense. The latter setting applies by default.

- `ErrorLog` specifies the filename of the log file for error messages.
- `CustomLog` specifies the file name of the access log. In this file, Apache logs each successful transfer of a file. To the second parameter, you must pass either the name of a predefined logging format or a string of your own format statements. The allowed format codes are described at *http://httpd.apache.org/docs/2.4/de/mod/mod_log_config.html*.

  On Ubuntu, some formats are preconfigured in `apache2.conf`, such as `combined` or `common`.
- `ErrorDocument` specifies how Apache should respond to errors. Enter the error number as the first parameter (e.g., 404 for *not found*) and the name of a local file or the address of an external page to be displayed in this case in the second parameter. The file name must be specified relative to `DocumentRoot`. The main error codes are as follows:
  - 400, bad request
  - 401, authorization required
  - 403, forbidden
  - 404, not found
  - 500, internal server error

  A list of all HTTP status codes can be found at *https://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html*.

  By default, `ErrorDocument` is not configured. To avoid unpleasant error messages, you should take the trouble to set up an error page and specify its location using `ErrorDocument`.
- `Alias` establishes a mapping between a web directory and a directory of the hard disk (even outside `DocumentRoot`). For example, `Alias /mytool /usr/local/mytool` causes the files from the `/urs/local/mytool` directory to be read when *http://myserver.com/mytool* is accessed.

  As a rule, you need to set the access rights for each `alias` directory in a `<Directory>` group (see the following section). The `ScriptAlias` variant of `Alias` is used to define directories with CGI scripts.

24.4.2 Directory Configuration

Following these settings, which apply to the entire virtual host, you can set the properties for individual directories of your host in one or more `<Directory "/directory/">` groups. The following list contains only the most important keywords:

- `DirectoryIndex` specifies which file Apache should send if an address ends with / and thus affects an entire directory (`index.html` by default).

Multiple files may also be specified. If that's the case, Apache works through all the entries in order up to the first hit (for example, `DirectoryIndex index.php index.html`).

- `Options` allows you to specify various options that apply to the directory. These include the following:
  - `ExecCGI`: Run CGI scripts
  - `FollowSymLinks`: Follow symbolic links
  - `Includes`: Add include files (`mod_include` module)
  - `Indexes`: Show file list when `index.html` is missing
  - `MultiViews`: Automatic language selection (`mod_negotiation` module)

  By default, Apache uses the `All` setting. This activates all options with the exception of `MultiViews`. The Ubuntu configuration is more restrictive: `Options FollowSymLinks` applies to the entire file system, and `Options Indexes FollowSymLinks MultiViews` applies to the `/var/www` directory.

  To deactivate individual options compared to the default settings of a parent directory, a minus sign must be placed in front of them. A preceding plus sign is also allowed but has no effect; the option is enabled exactly as if it were specified without a plus sign.

  For safety reasons, the "less is more" motto should prevail when it comes to `options`. The `Indexes` option tells curious web surfers the names of all files located in a directory if you forget `index.html` once. This is a potential security risk. You need `MultiView` only for multilingual websites with automatic language selection. If your site does not offer something like this, you can also do without this option.
- `AllowOverride` specifies which directory-specific settings may be changed by an `.htaccess` file. The following options are available:
  - `AuthConfig`: Set authentication method
  - `FileInfo`: Set file and document types
  - `Indexes`: Modify directory index
  - `Limit`: Change access rights (`Allow`, `Deny`, `Order`)
  - `Options`: Change directory options

  By default, all options are active in Apache; that is, each option can be changed. In most distributions, however, the default configuration is more restrictive due to security reasons. Thus, on Debian and Ubuntu, `None` is preset for all relevant directories (see `/etc/apache2/apache2.conf`).

24.4.3 Securing Directories

There is one central aspect that I have not yet addressed: the control of access rights for directories. Apache versions 2.2 and 2.4 differ significantly in this respect. Because the version 2.2 syntax still works on many Apache 2.4 installations and is widely used, I cover both variants here. For new installations, you must make sure to use the Apache 2.4 keywords!

Access rights for directories (Apache 2.2)

In Apache 2.2, you can use `Order`, `Allow`, and `Deny` within the `<Directory>` group to set the circumstances under which Apache may read and redirect files from the respective directory. Access rules also apply to all subdirectories, unless other rules are explicitly defined in a different `<Directory>` group. The access rules for the / directory therefore specify default rules for the entire file system:

- `Order Allow,Deny` means that first all `Allow` and then all `Deny` rules will be analyzed. If no rule can be applied to a page access, the access will be blocked.
- `Order Deny,Allow` reverses the order of the rules. However, you should note that if no rule fits for a page access, the access is allowed! This rule applies by default in Apache.
- `Allow from` specifies from which host names or IP addresses access is allowed, such as `Allow from 213.214.215.216 knownpage.com`. You can specify IP address ranges in the format 213.214 or 213.214.0.0/255.255.0.0 or 213.214.0.0/16 (for 213.214.*.*). For host names, `site.com` also applies to `www.site.com`, `sub.site.com`, and so on. The `Allow from all` rule allows every access.
- `Deny from` works just the other way around and blocks access for the specified hosts or addresses.

By default, `Order Deny,Allow` applies, and in the absence of other rules, all access to all directories is thus blocked! If you place web files in other directories, you must not forget to allow access to them.

Access rights for directories (Apache 2.4)

On Apache 2.4, the `Order`, `Allow`, and `Deny` keywords are considered obsolete. However, the keywords are still processed by the `mod_access_compat` module. This module is available on most Apache 2.4 installations and ensures that an Apache 2.4 update does not overwrite the entire previous configuration.

The use of the new `Require` keyword is recommended for a new configuration. The following examples show different types of use:

```
# allows access from the computer with IP address 192.168.0.2
Require ip 192.168.0.2
```

```
# allows access from the address range 10.0.*.*
Require ip 10.0

# allows access from an IPv6 address range
Require ip 2001:1234:789a:0471::/64

# allows access for a specific hostname
Require host intern.my-company.com

# allows access for *.my-company.com
Require host my-company.com

# allows access from localhost (IPv4 and IPv6)
Require local

# allows access for authenticated users
Require valid-user

# allows access from anywhere
Require all granted

# blocks any access
Require all denied
```

If you formulate several conditions for a `<Directory>`, then it is sufficient if *one* of these conditions is fulfilled:

```
<Directory /var/www/cms>
  Require local
  Require ip 192.168
  Require host my-company.com
</Directory>
```

With `<RequireAll>`, you can combine several conditions with a logical AND. Apache returns the requested page only if *all* conditions are true at the same time:

```
<Directory /var/www/internal-wiki>
  <RequireAll>
    Require valid-user
    Require ip 192.168.17
  </RequireAll>
</Directory>
```

24.4.4 Password Protection for Web Directories

It often happens that web directories can only be released after authentication with a user name and the corresponding password. Apache provides a simple method for this, which works equally well in versions 2.2 and 2.4.

Password file

The first step toward password protection is a password file. For security reasons, the file should be created outside all web directories to prevent access via a web address. The following example assumes that the password file is stored in the `/var/www-private` directory. If you set up a new directory, you should make sure that Apache has read permissions for it. On Fedora/RHEL, you must also keep SELinux in mind and either set up the password directory inside `/var/www` or set the SELinux context correctly after creating the directory.

To create a new password file, you can use the `htpasswd` command with the `-c` option (*create*). The password is of course encrypted:

```
root# cd /var/www-private
root# htpasswd -c passwords.pwd username
New password: ********
Re-type new password: ********
Adding password for user username
```

Additional username/password pairs can be added via `htpasswd` without the `-c` option:

```
root# cd /var/www-private
root# htpasswd passwords.pwd name2
New password: ********
Re-type new password: ********
Adding password for user username
```

Configuration

There are now two ways to configure Apache to actually honor the password file. The first variant requires that you carry out the configuration directly in an Apache configuration file—on Debian or Ubuntu in `/etc/apache2/sites-available/default` for the server's default website, or in `.../sitename` for a virtual host. In the second variant, the configuration is carried out in the `.htaccess` file located inside the web directory.

To ensure that the password file is taken into account by Apache, you must add various authentication options to the `<Directory>` group. If the directory you want to protect does not already have its own `<Directory>` group, you can create a new group. All options will automatically be taken over

from the parent directory. So you just need to add the authentication options. The following lines provide a pattern for this:

```
# in /etc/apache2/sites-available/xxx (Debian/Ubuntu)
...
# password protected directory
<Directory "/var/www/admin/">
    AuthType     Basic
    AuthUserFile /var/www-private/passwords.pwd
    AuthName     "admin"
    Require      valid-user
</Directory>
```

Let me briefly explain the keywords:

- `AuthType` specifies the authentication type. I only discuss the `Basic` type here.
- `AuthUserFile` specifies the location of the password file.
- `AuthName` indicates the realm for which the access is valid. The point is that you don't have to log in every time you want to access different directories that are protected by the same password file. Once you have logged in with a specific `AuthName` designation, this login also applies to all other directories with this `AuthName`.
- `Require valid-user` means that any valid username and password combination is allowed as login. Alternatively, you can also specify here that a login is only allowed for very specific users:

  ```
  Require user name1 name2
  ```

.htaccess file

The method outlined here is only possible if you have access to the Apache configuration files—that is, if you are the web administrator yourself. If that's not the case, equivalent protection can also be provided by the `.htaccess` file located in the directory to be protected. This file must contain the same statements that were specified earlier within the `<Directory>` group: `AuthType`, `AuthUserFile`, `AuthName`, and `Require`.

.htaccess Requires AllowOverride AuthConfig

`.htaccess` files are only noticed if a change of authentication information is allowed within the web directory. The (parent) `<Directory>` group must contain `AllowOverride AuthConfig` or `AllowOverride All`.

24.5 Virtual Hosts

A separate web server for each website (for each host) would be a waste of resources given the performance of current computers! Thanks to virtual hosts, a web server can operate many websites in parallel.

From a technical perspective, there are three methods for Apache to decide which virtual host a web request is directed to. In any case, the HTTP header transmitted from the browser to the server serves as the starting point:

- **Name-based virtual hosts**
  Apache recognizes the desired website by the host name contained in the HTTP header. This variant is the easiest to implement and the most widespread.
- **IP-based virtual hosts**
  Apache recognizes the required website by the IP address in the header. This approach is associated with a serious disadvantage: each virtual host requires its own IP address, and IP addresses are scarce for IPv4.
- **Port-based virtual hosts**
  Apache recognizes the required website based on the port number. This variant is unusual in practice because the port number must be specified as part of the web address. This looks confusing and is at best suitable for a technically savvy target group, such as administrators.

In this section, I again refer to the default configuration of Debian or Ubuntu. There, it is common to use a separate configuration file for each virtual host. If you set up your web server under Fedora or RHEL, the same Apache keywords will be used. However, all settings are made either in the central `/etc/httpd/conf/httpd.conf` file or in individual files for each host in `/etc/httpd/conf.d/sitename.conf`.

Name-based virtual hosts and HTTPS

In the past, it was impossible to combine HTTPS with virtual hosts that differed only by host name. The problem was that the host name itself was also transmitted encrypted. It was thus impossible for the web server to "guess" the correct certificate.

Meanwhile, web browsers send the host name once unencrypted beforehand when establishing a connection. This tells Apache which certificate it must use for encryption. For this to work, `SSLStrictSNIVHostCheck off` must apply, which is the default for current Apache versions. You can find more details on the correct configuration of multiple hosts, each of which uses its own HTTPS certificate, at *https://wiki.apache.org/httpd/NameBasedSSLVHostsWithSNI*.

24.5.1 Setting Up Virtual Hosts

On Debian and Ubuntu, the default website is defined as a virtual host in `/etc/apache2/sites-available/default`.

To define a new virtual host, you need to create a new file in the `/etc/apache2/sites-available/` directory. This file should contain exactly one `<VirtualHost>` group. The three listings ahead each give an example of a name-based, IP-based and port-based host. `ServerName` specifies the name of the host. This host name must be included in the header information of a web request for Apache to respond to it. Optionally, you can use `ServerAlias` to name additional names. For example, for the `ServerName www.myserver.com` setting, it is recommended to add `ServerAlias myserver.com` so that a virtual host can be used with or without the preceding `www.` letters.

Usually, you need *two* files or two settings groups within one file for each host. The first group applies to HTTP and usually contains only a redirect to HTTPS. The second group of options applies to HTTPS and contains, among other things, the SSL options with the location of the certificate:

```
# /etc/apache2/sites-available/example-com.conf (Debian/Ubuntu)
# for HTTP
<VirtualHost *:80>
    ServerName     example.com
    ServerAlias    www.example.com
    DocumentRoot   /var/www/html/example-com/
    Redirect       permanent / https://example.com/
</VirtualHost>
# for HTTPS
<VirtualHost *:443>
    ServerName     example.com
    ServerAlias    www.example.com
    DocumentRoot   /var/www/html/example-com/
    SSLCertificateFile      \
       /etc/acme-letsencrypt/example.com/fullchain.pem
    SSLCertificateKeyFile   \
       /etc/acme-letsencrypt/example.com/privkey.pem
    # ... other settings
</VirtualHost>
```

Enabling virtual hosts

To enable or later disable a virtual host, you should now run `a2ensite name` or `a2dissite name` on Debian/Ubuntu and then prompt Apache to reload the configuration files:

```
root# a2ensite example-com          (Debian/Ubuntu)
root# systemctl reload apache2
```

Theoretically, it's also possible to disable the server's default website using `a2dissite`. But you should not do that, because the `/etc/apache2/sites-available/000-default.conf` file often contains various default settings for Apache!

Once you set up virtual hosts, the default website defined in `sites-available/default` displays only if web requests do not apply to any of the virtual hosts.

On Fedora, RHEL, and SUSE, the `a2ensite/a2dissite` commands are omitted because all information about the virtual hosts is located in the central `httpd.conf` configuration file or in separate files in the `conf.d` directory. You can activate changes made there by using the following command:

```
root# systemctl reload httpd          (Fedora, RHEL)
root# systemctl reload apache2        (SUSE)
```

Logging

By default, all web accesses are stored in central logging files. Usually it's more convenient to provide separate logging files for each host. For this purpose, you must include the `ErrorLog` and `CustomLog` keywords in the two `VirtualHost` sections:

```
<VirtualHost *:80>
    ErrorLog    /var/log/apache2/example-com.error.log
    CustomLog   /var/log/apache2/example-com.access.log combined
</VirtualHost>
<VirtualHost *:443>
    ErrorLog    /var/log/apache2/example-com-ssl.error.log
    CustomLog   /var/log/apache2/example-com-ssl.access.log combined
</VirtualHost>
```

You can find more information about the many variants and keywords for logging configuration at *https://httpd.apache.org/docs/2.4/logs.html*.

24.6 Web Access Statistics

If you run your own web server or administrate it for someone else, you usually also want to know how many people visit the website per day, which web browsers are used, and so on. I'll briefly introduce a few tools here first, and then go into a bit more detail about using GoAccess.

Webalizer and AWStats

In the past, the Webalizer or AWStats program was often used to create such statistics. These programs from the early days of the web are no longer up to date. The configuration is relatively complex because calling them must be synchronized with the rotation of the logging files.

Google Analytics

The undisputed champion among modern web analyzer tools is Google Analytics. To be able to use it, you need to include JavaScript code on the pages of your website that redirects each page access to a central server. This approach makes it easier to distinguish between "real" visitors and search robots, and it produces more accurate results that can be observed in real time.

As Google Analytics does not analyze logging files but is based on code that is integrated directly into the website, Google Analytics copes much better with modern websites where you never leave the home page (single-page web applications).

Note, however, that the use of Google Analytics in compliance with data protection laws like GDPR is a problem! In any case, you need to include a confirmation dialog in your website that informs visitors about the use of Google Analytics. This dialog is often combined with the cookie consent form required in the European Union (albeit completely pointless; see *https://marketingplatform.google.com/about/analytics/*).

Matomo

Matomo (*https://matomo.org*; formerly Piwik) is an open-source alternative to Google Analytics. Its use avoids feeding Google with even more data; the data collected remains under your control. However, from a privacy perspective, Matomo acts similarly to Google Analytics; thus, even if you use Matomo, you should include a prominent notice on your website.

24.6.1 GoAccess

In my opinion, GoAccess (*https://goaccess.io*) is a good compromise between the outdated Webalizer and AWStats programs on the one hand and privacy-problematic tools like Google Analytics or Matomo on the other hand. GoAccess analyzes the logging files of the web server and in this respect acts similar to Webalizer or AWStats. However, unlike these programs, the program is also suitable for real-time analysis of web traffic. Moreover, it can be used in a terminal window via SSH if required, without the otherwise usual intermediate step of generating reports as web pages.

Installation

GoAccess is available as a package in many distributions and can be installed using `apt/dnf/zypper install goaccess`. On RHEL and its clones, you must enable the EPEL package source upfront.

Use in the terminal

The easiest way is to call `goaccess` in an SSH session and pass the filename of the Apache logging file as a parameter. On Ubuntu, even a nonprivileged user can do this:

```
user$ goaccess /var/log/apache2/access.log
```

On Fedora and RHEL, the logging files are located in `/var/log/httpd` and are only accessible to `root`. The call of `goaccess` must then be made with `root` privileges:

```
root# goaccess /var/log/httpd/access_.log
```

At startup, you may need to specify the format of the logging file. Usually, it is sufficient to confirm the default **NCSA Combined Log Format** entry. `goaccess` then displays an analysis of the logging file and updates it regularly until you exit the program using Q (see Figure 24.5).

You can use the cursor or Tab keys to scroll through additional sections (static accesses, 404 errors, host names and IP addresses, etc.). Enter adds more details to the currently active section. S sorts the rows according to a different criterion. A summary of other keyboard shortcuts is provided by H.

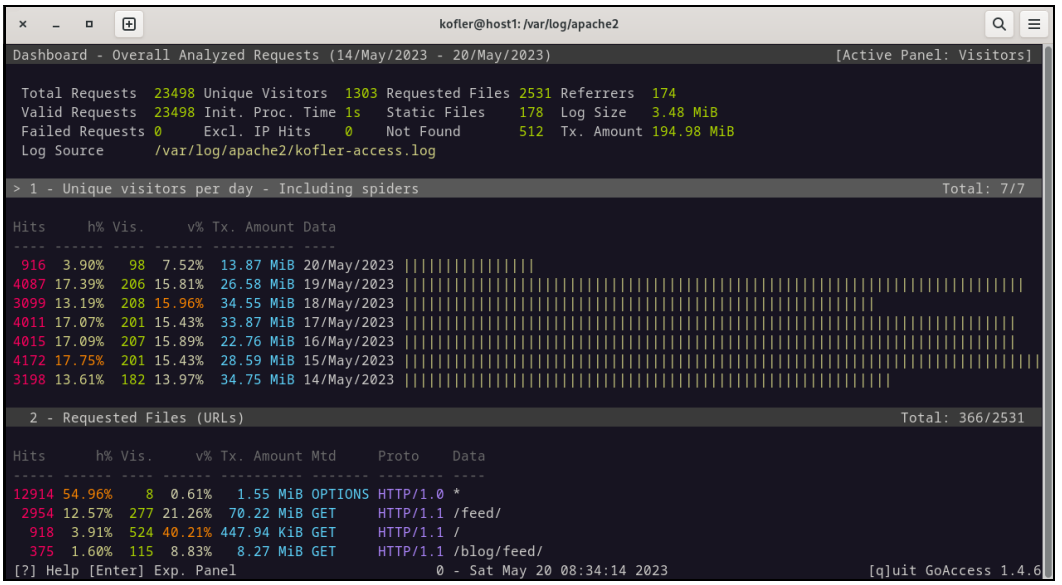


Figure 24.5 Access Statistics in Terminal Window

Most distributions set up new access log files daily or weekly and compress the older files (see also the description of `logrotate` in Chapter 14, Section 14.12.4). The following command processes all compressed files as well as the two uncompressed `access.log` and `access.log.1` files. Note that such a call of `goaccess` initially takes quite a long time (all logging files have to be decompressed and loaded) and that the memory requirements of `goaccess` in this case are considerable; depending on the size of the log files, this could be in the GiB range! If you replace the * character with ?, only the last ten logging files will be considered, which is often sufficient:

```
user$ cd /var/log/apache2
user$ zcat access.log access.log.*.gz | \
        goaccess --log-format=COMBINED \
        access.log access.log.1  -
```

Format error

If `goaccess` returns an error message of the type *No time format was found on your conf file*, you have two options: you can either specify the log format of your web server with an option (as in the above example with `--log-format`) or write the format in the configuration file whose location is shown in the context of the error message (often `/etc/goaccess.conf`).

Generating HTML reports

By additionally passing `goaccess` the `-o out.html` option, you can create an HTML file with the access statistics. You can then view this page in a web browser (see Figure 24.6). Of course, you can automate the generation of such access statistics in a Cron job and run it once a day or once a week:

```
user$ cd /var/log/apache2
user$ zcat access.log.?.gz | goaccess access.log access.log.1  \
        --log-format=COMBINED -o /var/www/html/myreports/out.html
```

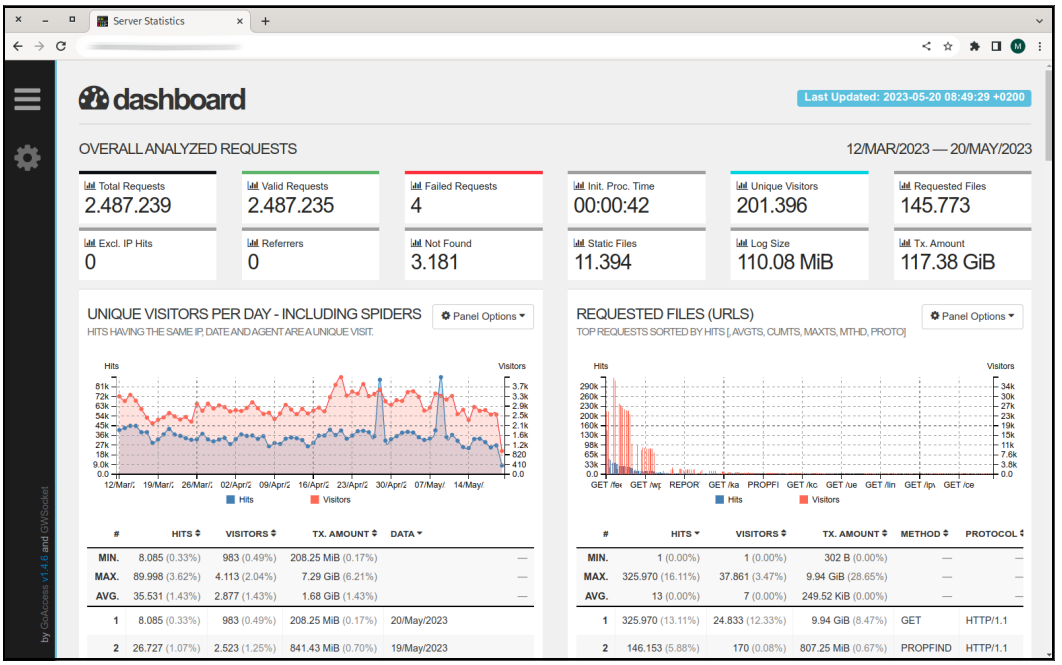


Figure 24.6 Analyzing Access Statistics in Web Browser

`goaccess` can even provide real-time updates on the HTML page. To do this, you must also pass the `--real-time-html` option to `goaccess`. If the resulting page is to be transferred over an HTTPS connection, you must also pass the

certificate and its key in parameters. Accessing these files requires running the command with root privileges:

```
root# cd /var/log/apache2
root# goaccess --log-format=COMBINED access.log access.log.1 \
        -o /var/www/html/myreports/uploads/report.html \
        --real-time-html \
        --ssl-cert=/etc/acme-letsencrypt/mysite.fullchain \
        --ssl-key=/etc/acme-letsencrypt/mysite.key
```

24.7 PHP

Dynamic web pages

Apache itself can only transfer static websites. However, all modern websites use dynamic pages. Each time such a page is requested, Apache starts an external program, processes the code of the page, and delivers a customized page as a result. Thus the page may contain, for example, the current time, the result of a database query, a constantly changing advertisement, and so on.

PHP

Countless programming languages can be used for programming dynamic websites, such as PHP, Python, Java, or JavaScript. The basic idea of a PHP web page is that the file with the *.php identifier contains both HTML and PHP code. PHP code is introduced with the <?php tag and ends with ?>.

When a web user requests a PHP page, Apache passes the page to the PHP interpreter where the PHP code will be executed. The result of the code is embedded directly into the HTML file. The PHP interpreter returns the resulting page to Apache, and Apache sends it to the web user. So the user's web browser never sees the PHP code, only the resulting HTML page.

Hello, World!

There is not enough space here for an introduction to the PHP programming language. Instead, the following small example is intended to illustrate the concept of PHP. The following file provides an HTML page with the current time after it has been processed by the PHP interpreter:

```
<!DOCTYPE html>
<html><head>
  <meta http-equiv="Content-Type"
        content="text/html; charset=utf-8" />
  <title>PHP Example</title>
</head><body>
<p>The current time on this server:

<?php
  date_default_timezone_set('America/Detroit');
```

```
  echo strftime('%Y-%m-%d %H:%M:%S');
  echo "<p>Special character test: äöü";
?>
</body></html>
```

Installation

If PHP has not already been installed together with Apache, you can install the required `php` packages using your package management program. What is "required," however, is not so easy to determine.

Similar to Apache, PHP is also distributed across numerous packages that contain the language itself as well as various extensions. For first experiments, the following commands can serve as a starting point:

```
root# dnf install php php-fpm php-gd php-mbstring  (Fedora, RHEL)
root# apt install php php-gd php-mbstring          (Debian, Ubuntu)
root# zypper install -t pattern lamp_server        (SUSE)
```

Unless package management takes care of this, you must restart Apache after the installation so that the web server takes newly added PHP modules into account:

```
root# systemctl enable --now php-fpm               (Fedora, RHEL only)
root# systemctl restart apache2/httpd              (all distributions)
```

Latest PHP packages for RHEL

Because of the long life of RHEL, this distribution and its clones offer parallel PHP packages for multiple versions. When I revised this chapter in March 2024, only PHP version 8.1 was available for RHEL 9. On RHEL 8, on the other hand, the originally delivered version 7.2 and three newer versions were available for selection. In this example, I therefore refer to RHEL 8, where the concept of the module system is easier to recognize:

```
root# dnf module list php
dnf module list php
AlmaLinux 8 - AppStream
Name  Stream   Profiles                    Summary
php   7.2 [d]  common [d], devel, minimal  PHP scripting language
php   7.3      common [d], devel, minimal  PHP scripting language
php   7.4      common [d], devel, minimal  PHP scripting language
php   8.0      common [d], devel, minimal  PHP scripting language
```

To upgrade to the latest PHP version, you should proceed as follows:

```
root# dnf module reset php
root# dnf module enable php:8.0
root# dnf install php php-fpm php-gd php-mbstring
root# systemctl enable --now php-fpm
root# systemctl restart httpd
```

Remi package source

Compared to the past, it is great that there are official PHP modules for multiple versions. But you still have to wait (many) months for the latest PHP version. For example, when I revised this chapter, PHP 8.2 had been released a year ago and PHP 8.3 had been around for a month. `dnf module list php` didn't know about both versions, neither in RHEL 8 nor in RHEL 9.

If you need the latest PHP version, it's best to turn to the Remi package source, which has earned an excellent reputation in this regard over the past few years. The following commands activate the Remi package source and the CRB and EPEL sources that are also required. The first three commands apply to RHEL, while the other three commands apply to RHEL clones:

```
root# subscription-manager repos --enable \                (RHEL)
  codeready-builder-for-rhel-9-x86_64-rpms
root# dnf install \
  https://dl.fedoraproject.org/pub/epel/epel-release-latest-9.noarch.rpm
root# dnf install https://rpms.remirepo.net/enterprise/remi-release-9.rpm

root# dnf config-manager --set-enabled crb                 (clones)
root# dnf install epel-release
root# dnf install https://rpms.remirepo.net/enterprise/remi-release-9.rpm
```

`dnf module list php` now also lists modules from the Remi package source:

```
root# dnf module list php
AlmaLinux 9 - AppStream
Name    Stream     Profiles                     Summary
php     8.1        common [d], devel, minimal   PHP scripting

Remi's Modular repository for Enterprise Linux 9 - x86_64
Name  Stream    Profiles                     Summary
php   remi-7.4  common [d], devel, minimal   PHP scripting language
php   remi-8.0  common [d], devel, minimal   PHP scripting language
php   remi-8.1  common [d], devel, minimal   PHP scripting language
php   remi-8.2  common [d], devel, minimal   PHP scripting language
```

Now you can also use PHP 8.2:

```
root# dnf module reset php
root# dnf module enable php:remi-8.2
```

Run the remaining commands for installing and restarting the web server as described previously.

Configuration Countless options of the PHP interpreter are controlled by the `php.ini` file. As a rule, you can simply keep the basic settings.

The location of this file as well as other PHP configuration files once again depends on the distribution:

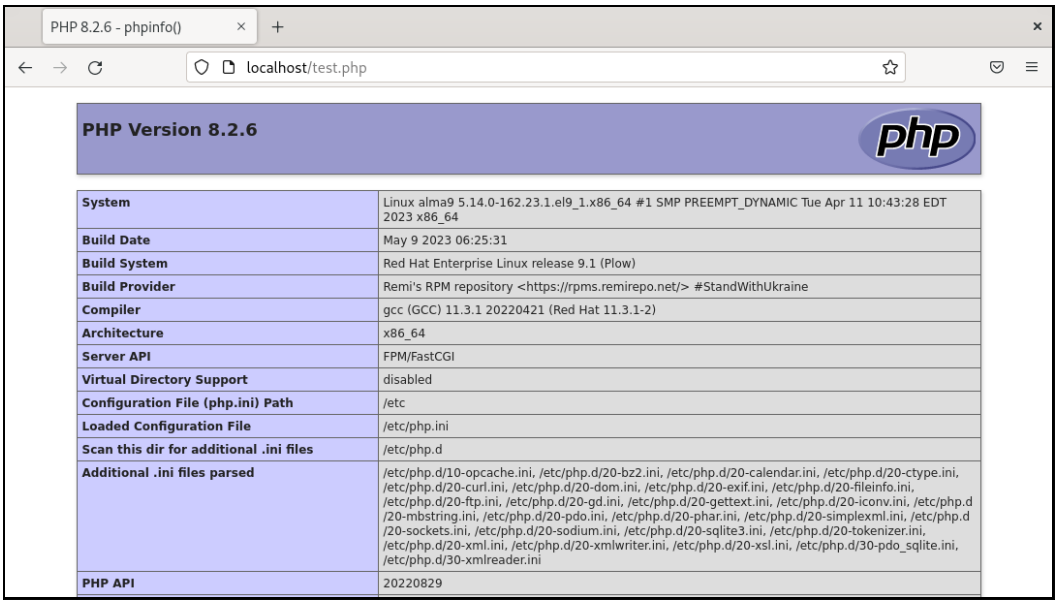
- Debian and Ubuntu:`/etc/php/<n>/apache2/php.ini`, `/etc/php/<n>/apache2/conf.d/*.ini`
- Debian and Ubuntu with FPM/FastCGI: `/etc/php/<n>/fpm/php.ini`, `/etc/php/<n>/fpm/conf.d/*.ini`
- Fedora, RHEL: `/etc/php.ini`, `/etc/php.d/*.ini`
- SUSE: `/etc/php<n>/apache2/php.ini`

Test To test whether the PHP installation works, you can create the `test.php` file, which consists of only a single line of code:

```
<?php phpinfo(); ?>
```

Copy this file to the `DocumentRoot` directory (see Table 24.1), and make sure that Apache is allowed to read the file. Although PHP files are actually script files, read permissions are sufficient. Access rights for execution (x-access bits) are not required.

Using a web browser, you can then view the *http://localhost/test.php* or *https://<hostname>/test.php* page. The result is a very extensive page containing all possible options and settings of Apache and PHP (see Figure 24.7).



System	Linux alma9 5.14.0-162.23.1.el9_1.x86_64 #1 SMP PREEMPT_DYNAMIC Tue Apr 11 10:43:28 EDT 2023 x86_64
Build Date	May 9 2023 06:25:31
Build System	Red Hat Enterprise Linux release 9.1 (Plow)
Build Provider	Remi's RPM repository <https://rpms.remirepo.net/> #StandWithUkraine
Compiler	gcc (GCC) 11.3.1 20220421 (Red Hat 11.3.1-2)
Architecture	x86_64
Server API	FPM/FastCGI
Virtual Directory Support	disabled
Configuration File (php.ini) Path	/etc
Loaded Configuration File	/etc/php.ini
Scan this dir for additional .ini files	/etc/php.d
Additional .ini files parsed	/etc/php.d/10-opcache.ini, /etc/php.d/20-bz2.ini, /etc/php.d/20-calendar.ini, /etc/php.d/20-ctype.ini, /etc/php.d/20-curl.ini, /etc/php.d/20-dom.ini, /etc/php.d/20-exif.ini, /etc/php.d/20-fileinfo.ini, /etc/php.d/20-ftp.ini, /etc/php.d/20-gd.ini, /etc/php.d/20-gettext.ini, /etc/php.d/20-iconv.ini, /etc/php.d/20-mbstring.ini, /etc/php.d/20-pdo.ini, /etc/php.d/20-phar.ini, /etc/php.d/20-simplexml.ini, /etc/php.d/20-sockets.ini, /etc/php.d/20-sodium.ini, /etc/php.d/20-sqlite3.ini, /etc/php.d/20-tokenizer.ini, /etc/php.d/20-xml.ini, /etc/php.d/20-xmlwriter.ini, /etc/php.d/20-xsl.ini, /etc/php.d/30-pdo_sqlite.ini, /etc/php.d/30-xmlreader.ini
PHP API	20220829

Figure 24.7 PHP Test Page

For security reasons, it is not recommended to place such a page freely accessible on the internet. It contains a lot of information about the configuration of your web server.

Enabling FPM/FastCGI

The Apache web server can process multiple web requests simultaneously. Depending on the configuration, there are different *multiprocessing modules* (MPMs) that are responsible for this. In the past, it was common practice to use the `prefork` method to process PHP code. On Debian 12 and Ubuntu 22.04, this method is still used by default.

For reasons of speed, however, it is now recommended to switch to the much more efficient FPM/FastCGI method. On RHEL 8, it is already used by default if the `php-fpm` package is installed. You can verify this by looking at the PHP test page (see Figure 24.7; **Server API = FPM/FastCGI**).

To switch Ubuntu 22.04 from the `prefork` module to the FPM/FastCGI method, you need to run the following commands. The same procedure applies to Debian 12. However, there you have to replace the version number 8.1 with 8.2:

```
root# apt install php-fpm
root# apt remove libapache2-mod-php8.1
root# a2enmod proxy_fcgi setenvif
root# a2enconf php8.1-fpm
root# systemctl restart apache2
root# systemctl restart php8.1-fpm
```

If it does not work

If you see the PHP code instead of the test page or are offered the PHP file for download, the PHP code was not executed. Did you restart Apache after installing PHP or changing configuration files?

If it has not worked once, the cache of your web browser can then throw a spanner in the works: instead of requesting the page again from Apache, which might work now, the browser reads the page from the internal cache. To be on the safe side, you should restart the browser or delete the cache!

PHP application installation

Before you can install PHP programs like WordPress, you usually still need to set up the MySQL or MariaDB database server. Examples for installing web applications therefore follow a bit later (see Chapter 25, Section 25.4).

24.8 NGINX

Where possible, I try to focus on *one* program at a time in this book—especially in the server section. With web servers, this is increasingly difficult: the NGINX program now has a higher market share than Apache.

The popularity of NGINX results from the fact that it delivers static web pages more efficiently than Apache. However, for websites implemented with Java, JavaScript, PHP, and the like, the speed advantage is largely invalid. The largest share of the computing time is now taken up by the respective programming language and the database server that supplies the dynamically generated content.

Even though I still believe that Apache is the better starting point for beginners, I want to show in this section that using NGINX is not witchcraft either. For reasons of space, however, I cannot go into as many special cases, variants, and options here as I did with Apache. If necessary, you should take a look at the NGINX website, which provides comprehensive documentation:

- *https://nginx.org/en/docs*
- *https://www.nginx.com/resources/wiki/start*

LAMP and LEMP

The combination of Linux, Apache, MySQL or MariaDB, and PHP is usually called LAMP. If Apache is now replaced by NGINX, the result is actually LNMP. Instead of this unpronounceable abbreviation, LEMP has become common, referring to the usual pronunciation of NGINX as *Engine X*.

Installation on Ubuntu

You can use the following command to install NGINX and PHP on Ubuntu:

```
root# apt install nginx php-fpm
```

Then you need to change two configuration files. The required statements are mostly already there; you just need to remove the comment sign and change the setting if necessary. Depending on the distribution, you may need to specify more recent PHP version numbers instead of `8.1`. Also, of course, you need to replace the sample host name `lemp.a-company.com` with your actual domain name:

```
# file /etc/php/8.1/fpm/php.ini
...
cgi.fix_pathinfo=0
date.timezone="Europe/Berlin"

# file /etc/nginx/sites-available/default
server {
  # set domain name
  server_name lemp.a-company.com;
  # automatically consider index.php
  index index.php index.html index.htm index.nginx-debian.html;
```

```
  # process PHP scripts with FPM/FastCGI
  location ~ \.php$ {
    include snippets/fastcgi-php.conf;
    fastcgi_pass unix:/var/run/php/php8.1-fpm.sock;
  }
  ...
}
```

Problems with Backup Files

On Debian and Ubuntu, you need to make sure that there are no backup files of your editor in `/etc/nginx/sites-enabled`! NGINX also analyzes the backup files and then complains about duplicates in the configuration.

Finally, you must restart the web server:

```
root# systemctl restart php8.1-fpm
root# systemctl restart nginx
```

For testing, save the `<?php phpinfo(); ?>` statement in `/var/www/html/test.php` and check in the web browser whether *a-company.com/test.php* works. The resulting page should look like the one shown in Figure 24.7.

Let's Encrypt

To install the certificates from Let's Encrypt, it's best to use `acme.sh` (see Section 24.3). When executing `acme.sh --install-cert`, you must ensure that you adapt the option in compliance with NGINX:

```
root# acme.sh --install-cert ... --reloadcmd 'systemctl restart nginx'
```

SSL configuration

Finally, you need to adjust the NGINX configuration. The `ssl_certificate` and `ssl_certificate_key` keywords point to the location of the certificates generated by `acme.sh`. You can take the other options for SSL configuration from the website at *https://ssl-config.mozilla.org*:

```
# e.g. file /etc/nginx/sites-available/default
# or       /etc/nginx/sites-available/lemp.a-company.com
server {
  ...
  # use HTTPS
  listen 443 ssl http2;
  listen [::]:443 ssl http2;

  # certificates
  ssl_certificate /etc/acme-letsencrypt/lemp.a-company.com.fullchain;
  ssl_certificate_key /etc/acme-letsencrypt/lemp.a-company.com.key;
```

```
  # SSL options
  ssl_protocols TLSv1.2 TLSv1.3;
  ssl_ciphers ECDHE-ECDSA-AES128-GCM-SHA256:ECDHE-RSA-...;
  ssl_prefer_server_ciphers off;
  add_header Strict-Transport-Security "max-age=63072000" always;
  ssl_stapling on;
  ssl_stapling_verify on;
  ssl_trusted_certificate \
    /etc/acme-letsencrypt/lemp.a-company.com.fullchain;
}
server {
  listen              80 default_server;
  listen              [::]:80 default_server;
  # redirect HTTP -> HTTPS
  if ($host = lemp.a-company.com) {
      return 301 https://$host$request_uri;
  }
  ...
}
```

Securing directories

Securing individual directories with a password basically works in the same way as with Apache. You must install the `apache2-utils` package, which contains the `htpasswd` command described earlier in this chapter. Using this command, you can set up a password file—for example, like this:

```
root# htpasswd -c /etc/nginx/.htpasswd <username>
New password:  *********
Re-type new password:  *********
```

In the configuration file, you instruct NGINX to use this password file for a directory:

```
# file /etc/nginx/sites-available/default
server {
  location /phpmyadmin {
    index index.php;
    ...
    auth_basic            "Login required";
    auth_basic_user_file  /etc/nginx/.htpasswd;
  }
}
```

Logging

In the default configuration of Ubuntu, NGINX logs all page accesses in `/var/log/nginx/access.log` and `error.log` (see the `/etc/nginx/nginx.conf` configuration file).

Chapter 25
MySQL and MariaDB

MySQL has been the most popular database system in the open-source world for many years. Even if you don't plan to manage any databases yourself, there are countless web applications that require MySQL Server—for example, wikis, discussion forums, content management systems, or online stores.

This chapter describes how you can install the MySQL or MariaDB server on your computer and how to perform basic administration tasks. To provide a practical reference, the last section shows you how to install WordPress. The section is less about WordPress as such; rather, it is intended to illustrate the specific use of a LAMP/LEMP system.

However, I will not go into the design of databases, the use of SQL for querying or manipulating data, or the development of database applications; that would go beyond the scope here.

MySQL versus MariaDB

MySQL was originally developed by MySQL AB, a Swedish company, which was acquired by Sun. Later Oracle acquired Sun, and so now Oracle is the owner of MySQL. Some of the founders of MySQL later initiated a new project, the result of which is a largely MySQL-compatible database server called MariaDB.

The cooperation between Oracle and various Linux distributors was not smooth in the first years. In particular, many distributors complained that they did not receive information about security issues, or received it too late, making it impossible to quickly fix security vulnerabilities. This annoyance led to a number of distributions relying entirely on MariaDB. This is still true today for Debian, Fedora, and openSUSE, for example. The situation is different for distributions that are intended for commercial use: RHEL and Ubuntu provide packages for MySQL and MariaDB in the same way.

Older versions of MySQL and MariaDB were almost completely compatible with each other, so you could leave the database files unchanged when you switched to one or the other. However, this is no longer true for current versions: both database servers go their separate ways when it comes to additional functions, extensions, and special configuration settings. As soon as you use one of these extensions—perhaps unknowingly—it becomes possible to switch from one system to another only with some manual work.

Nevertheless, in real life most Linux users will not even notice whether the CMS, wiki, or other application is communicating with an original MySQL server or with a MariaDB server behind the scenes. In addition, the configuration files and administration tools and techniques described in this chapter apply equally to MySQL and to MariaDB. When I write *MySQL* in this chapter without further explanation, then this information is equally true for MySQL and for MariaDB.

License

MySQL and the associated drivers for various programming languages are subject to the GPL. Company-internal use of MySQL, use on a web server, and use in GPL projects are basically free of charge. Note, however, that the distribution of commercial projects (closed-source, no GPL) built on MySQL requires a commercial license for the MySQL server! Details about the license terms of MySQL can be found at *https://mysql.com/about/legal*.

MariaDB can only be used in accordance with the rules of the GPL.

25.1 Installation and Commissioning

Installation

MySQL or MariaDB packages are available for all common distributions. Some distributions even provide packages for both variants. The database server itself, its libraries, and the administration tools are in different packages. Table 25.1 summarizes which packages you usually need (as of spring 2024). You can install these packages using `apt`, `dnf`, or `zypper` depending on the distribution. For this book, I tested MySQL 8 on RHEL 8 and on Ubuntu 22.04, as well as MariaDB 10.11 on Debian 12.

In spring 2023, a number of distributions offered only very old MariaDB packages.

The installation of these packages is not recommended. It's better to set up an official MariaDB or MySQL package source first and then install the latest version. These package sources are often better maintained than the package sources of the distributions. Take a look at the following two pages:

- *https://mariadb.org/download/?t=repo-config*
- *https://dev.mysql.com/downloads/repo*

Also note that MariaDB distinguishes between long-term and short-term versions. Long-term versions are maintained for five years (currently applies to 10.5, 10.6, and 10.11), short-term versions only for one year. If you plan to use MariaDB for several years, you should choose the latest long-term version currently available.

Distribution	Database System	Packages
RHEL 9	MySQL 8	`mysql`, `mysql-server`
RHEL 9	MariaDB 10.5	`mariadb`, `mariadb-server`
Debian 12	MariaDB 10.11	`mariadb-client`, `mariadb-server`
Fedora 39	MariaDB 10.5	`mariadb`, `mariadb-server`
Fedora 40	MariaDB 10.11	`mariadb`, `mariadb-server`
openSUSE 15.5	MariaDB 10.6	`mariadb`, `mariadb-client`
Ubuntu 22.04 LTS	MySQL 8.0	`mysql-client`, `mysql-server`
Ubuntu 22.04 LTS	MariaDB 10.6	`mariadb-client`, `mariadb-server` (`universe` package source)

Table 25.1 MySQL and MariaDB in Package Sources of Various Distributions

Start/stop

MySQL or MariaDB is a daemon that must be explicitly started on Fedora, RHEL, and openSUSE (see Chapter 10, Section 10.5):

```
root# systemctl enable --now mysqld
root# systemctl enable --now mariadb
```

Database and logging files

Whether you use MySQL or MariaDB, the database files are always stored in the `/var/lib/mysql` directory.

A database server has various logging files for different tasks: error logging, transaction logging, update logging, and so on. If problems occur during startup, such as due to incorrect configuration, the error messages are recorded in the error log. The location of this file again varies greatly depending on the distribution (see Table 25.2).

Distribution	Location of Logging Files
Fedora/RHEL	`/var/log/mariadb/mariadb.log` or `/var/log/mysql/mysqld.log`
Debian	Journal
openSUSE	`/var/log/mysql/mysqld.log`
Ubuntu	`/var/log/mysql/error.log`

Table 25.2 Location for Error Logging in Different Distributions

In some distributions, you can also read MySQL or MariaDB status messages and errors in the journal:

```
root# journalctl -u mysqld or journalctl -u mariadb
systemd[1]: Starting mariadb.service - MariaDB 10.11.2 database
server...
mariadbd Starting MariaDB 10.11.2-MariaDB-1 source revision  as
process 14461
mariadbd InnoDB: Compressed tables use zlib 1.2.13
mariadbd InnoDB: Number of transaction pools: 1
mariadbd InnoDB: Using crc32 + pclmulqdq instructions
...
```

Configuration

The MySQL or MariaDB server is configured using the `/etc/my.cnf`, `/etc/mysql/my.cnf` files or a `*.cnf` file in a subdirectory within `/etc` (see Table 25.3).

Distribution	Location of the Server Configuration File
Fedora/RHEL	`/etc/my.cnf.d/mariadb-server.cnf` or `/etc/my.cnf.d/mysql-server.cnf`
Debian	`/etc/mysql/mariadb.conf.d/50-server.cnf`
openSUSE	`/etc/my.cnf`
Ubuntu	`/etc/mysql/mysql.conf.d/mysqld.cnf`

Table 25.3 Location for Database Server Configuration Files in Different Distributions

The `*.cnf` files are preconfigured for ordinary operation. For reasons of space, I cannot describe all the keywords for this file in detail. However, I want to pick out a few of them.

Basically, the configuration file is divided into several sections by `[name]`. Ahead, I refer exclusively to the `[mysqld]` section, which concerns the MySQL or MariaDB server itself.

Changes to the `[mysqld]` section do not take effect until you restart the database server. The other sections in `my.cnf` or in the remaining configuration files are used to configure various client programs:

- **`bind-address = 127.0.0.1`**
  This setting makes sure that network connections to the database server are possible only from the local computer, not from other computers on the local network or from the internet. If MySQL/MariaDB is to be used only by local programs anyway, for example, by PHP scripts of the web server installed on the same computer, this setting increases the level of

security. Debian and Ubuntu have this setting by default. In most other distributions this setting is missing, but in some cases port 3306 is blocked by a firewall.

- **skip-networking**
  This setting prevents any network access to the MySQL or MariaDB server. Even local network connections are forbidden. The setting is even more restrictive than `bind-address = 127.0.0.1`. A connection can only be established for local programs that communicate with the database server through a socket file. This is true for PHP scripts and C programs, for example. Programs that communicate with the database server via TCP/IP cannot use the database server. This restriction affects all Java programs in particular. For this reason, `bind-address = 127.0.0.1` is usually more useful.
- **character-set-server and collation-server**
  For compatibility reasons, older MySQL and MariaDB server versions still use the Latin-1 character set and the Swedish (!) sorting order. You can check this as follows:

```
mysql> show variables like 'character_set_server';
latin1
mysql> show variables like 'collation_server';
latin1_swedish_ci
```

  The following two lines in the `[mysqld]` section of the configuration files provide a workaround for new databases:

```
character_set_server=utf8mb4
collation_server=utf8mb4_general_ci
```

IPv6

MySQL and MariaDB support IPv6. However, the database server is preconfigured by default to reject IPv6 connections. If you want to allow both IPv4 and IPv6 connections, you must use the `bind-address=::` setting in `my.cnf`. `bind-address=::1` allows only IPv6 connections through `localhost`. Note that an IPv6 connection can only be established if you have previously authorized the respective IP address using the `GRANT` SQL command or by making a direct change to the `host` column of the `mysql.user` table.

25.1.1 Access Protection

In current MySQL and MariaDB versions, there are two different authentication methods for logging in to the database:

- **Combination of name, host, and password**
  Three pieces of information are evaluated when the connection is established: the login name, the name of the host from which the connection

is established (often `localhost`), and finally a password. If all three pieces of data match a combination stored in MySQL, access is generally permitted.

It should be noted that the MySQL login names are managed completely separately from those of the Linux system. However, there is a default user whose name you are familiar with: the `root` user on MySQL has unlimited rights, just like the `root` user on Linux.

- **Connection to a Linux account**
  In this variant, there is a direct mapping between the name of a Linux account and that of a MySQL user. This assignment is made particularly often for `root`. In this case, whoever is logged in as `root` on the Linux machine may also administrate the MySQL server as `root` without any further login.

  Behind the scenes, the `auth_socket` plugin is responsible for this type of authentication on MySQL, and the `unix_socket` plugin on MariaDB.

The question as to which authentication method is used depends on the contents of the *user* table of the *mysql* database. The *mysql* database is set up by default with every installation of a MySQL or MariaDB server in order to save various server settings there.

The *user* table of this database on MySQL determines who can log on to the database server and how. In MariaDB, on the other hand, *user* is a view that shows the data stored in the new *global_priv* table.

MySQL versus mysql versusmysql

The term *MySQL* has different meanings depending on the context, which is why I format the terms differently. MySQL in this notation refers to the database server.

mysql in italics denotes a database in which the MySQL or MariaDB server stores various metadata—including the access rights mentioned earlier.

`mysql` is a command or client that can connect to the database server. As soon as a connection is established, SQL commands can be executed using `mysql`. `mysql` is often used for administrative tasks.

25.1.2 Securing MySQL/MariaDB

The default protection of the MySQL or MariaDB server depends strongly on the respective distribution. This section describes the major variants and refers to the packages from the distribution package sources.

Debian/Ubuntu with MySQL/ MariaDB

On current versions of Debian and Ubuntu, the database server is preconfigured in such a way that the local Linux user `root` is also the database administrator. `root` can connect to the MariaDB server without a password (authentication method 2). To try out the authentication process, you first need to log in to Linux as `root` and then run the `mysql` command without any other options:

```
root# mysql
Server version: 10.11.2-MariaDB-1 Debian
...

MariaDB> exit
```

Users other than `root` cannot connect to the MySQL or MariaDB server. So the default configuration is secure. As you will see momentarily, this is anything but self-evident!

RHEL 9 with MySQL 8

For a MySQL 8 installation on RHEL 8, the bleak motto is: *insecure by default*. Whoever can log on to a RHEL server gets full admin rights for the MySQL server in the subsequent step using `mysql -u root` without any password. The `mysql_secure_installation` command is provided for security purposes. The following (abbreviated) listing demonstrates its use:

```
root# mysql_secure_installation
Would you like to setup VALIDATE PASSWORD component? y
There are three levels of password validation policy:
  LOW    Length >= 8
  MEDIUM Length >= 8, numeric, mixed case, and special characters
  STRONG Length >= 8, numeric, mixed case, special characters,
                  dictionary file
  Please enter 0 = LOW, 1 = MEDIUM and 2 = STRONG: 1
Please set the password for root here.
  New password: ************
  Re-enter new password: ************
Disallow root login remotely? y
Remove test database and access to it? y
Reload privilege tables now? y
```

RHEL 9 with MariaDB 10.5

The MariaDB version shipped with RHEL 9 is outdated, but at least it is properly secured. Only the Linux users `root` and `mysql` are allowed to log in without a password. (Usually, the `mysql` user does not exist, which leaves only `root`.) This means that the same security rules apply as on Debian and Ubuntu.

Fedora and openSUSE

In current versions of Fedora and openSUSE, MariaDB is also secured, just like on Debian, RHEL, or Ubuntu: only the `root` user can log in without a password and is granted unrestricted access to the database server.

Other installations

Regardless of what distribution you use or from which package source you get MySQL or MariaDB, you can run the designated scripts to secure it:

```
root# mysql_secure_installation      (for MySQL)
root# mariadb-secure-installation    (for MariaDB)
```

25.1.3 Checking the Protection

MySQL and MariaDB have diverged greatly in recent years as far as login security is concerned. For outdated MySQL versions up to 5.5 and for just as old MariaDB versions up to 10.3, the authentication data is stored in four columns of the *mysql.user* table. Make sure that there is no user for whom the `password` and `plugin` columns are empty!

MySQL 5.7 and higher

In MySQL versions 5.7 and higher, the `password` column was replaced by `authentication_string`. There are no users on the following system who are allowed to log in without a password. When checking, make sure that the `authentication_string` column is not empty if the `plugin` column contains `mysql_native_password` or `caching_sha2_password`:

```
user$ mysql -u root -p
MySQL> SELECT user, host, authentication_string, plugin FROM
mysql.user;
   user             host       authentication_string     plugin
  ---------------  ---------  ------------------------  ------------
  mysql.infoschema  localhost  $A$005$THIS...  caching_sha2_password
  mysql.session     localhost  $A$005$THIS...  caching_sha2_password
  mysql.sys         localhost  $A$005$THIS...  caching_sha2_password
  root              localhost  *F0075F11B2...  mysql_native_password
  testuser          localhost  *462366917E...  mysql_native_password
```

MariaDB 10.4 and higher

In MariaDB, all authentication information is stored in the new *mysql.global_priv* table. This table contains a JSON string in the `Priv` column, which unfortunately does not tell you at a glance whether the configuration is secure or not. Accounts where this column only contains `{}` are definitely unsafe! The following installation is secure; only the `root` and `mysql` Linux users are allowed to log in without a password (`plugin = unix_socket`):

```
root# mysql
MariaDB> SELECT CONCAT(user, '@', host, ' => ', JSON_DETAILED(priv))
         FROM mysql.global_priv;
```

```
root@localhost => {
  "access": 18446744073709551615,
  "plugin": "mysql_native_password",
  "authentication_string": "invalid",
  "auth_or":
  [ {},
    {
          "plugin": "unix_socket"
    }
  ]
}
mysql@localhost => { analog ... }
```

25.1.4 Setting Up New Users

root user with password

In a configuration that links the `root` login for the database with a `root` login on the Linux system, it is often useful to provide a second administrator login for `root2` that is secured with a password (authentication method 1). To do this, you want to connect to the server again using `mysql` and then run the following SQL command:

```
root# mysql
MariaDB> CREATE USER root2@localhost IDENTIFIED BY 'secret';
MariaDB> GRANT ALL ON *.* TO root2@localhost WITH GRANT OPTION;
MariaDB> exit
```

Be sure to specify `root2` and not `root`; otherwise, you will overwrite the default `root` configuration. You can then try the `root2` login under any user account:

```
user$ mysql -u root2 -p
Enter password: *********    (password as for the CREATE-USER command)
```

You Should Use Different Passwords for Linux root and MySQL root!

The passwords of the Linux system and the passwords of the MySQL or MariaDB server are managed separately. No matter what distribution and database server you are running, for security reasons, you must never use the same passwords in the two systems!

By default, MySQL or MariaDB uses decidedly poor hash algorithms to encrypt passwords. If you have synchronized passwords, a security problem in the database server could also reveal the `root` password for the Linux account to the attacker.

Setting up users for a database

As on Linux, you should only work as root within MySQL or MariaDB in order to perform admin tasks. To access individual databases, it's best to set up separate MySQL accounts for each. To do this, log in as MySQL root and then run an SQL command according to the following pattern:

```
root# mysql [-u root -p]
mysql/
MariaDB> CREATE USER username@localhost IDENTIFIED BY 'topSecret';
mysql/MariaDB> GRANT ALL ON dbname.* TO username@localhost;
```

Of course, you need to adjust the database name, the account name, and the password!

25.1.5 First Tests

New database, new user

To test MySQL, you need to know the database language SQL, which I assume here. The goal of the following commands is to create the new database named mydatabase and a new user called newuser who is allowed to access this database. The easiest way to do this is to use the mysql program. It has a function that is comparable to a Linux shell: It receives SQL commands, passes them on to the MySQL or MariaDB server and finally displays the result. All SQL commands must end with a semicolon:

```
user$ mysql -u root -p
Enter password: ******
...
mysql> CREATE DATABASE mydatabase;
mysql> CREATE USER newuser@localhost IDENTIFIED BY 'xxxxxxxxx';
mysql> GRANT ALL ON mydatabase.* TO newuser@localhost;
mysql> exit
```

All subsequent tests with the new database can now be carried out by the MySQL user, newuser. As on Linux, it's also advisable to work as little as possible as root in MySQL.

Creating a table and filling it with data

The following commands are used by newuser to create a new table (CREATE TABLE), insert some data records (INSERT), and finally view all data records (SELECT). In the table, the id column plays a special role; the MySQL server independently inserts a unique number there for each new record. This number is used to identify the record:

```
user$ mysql -u newuser -p
Enter password: ******
mysql> USE mydatabase;
mysql> CREATE TABLE mytable (
         id INT NOT NULL AUTO_INCREMENT,
```

```
        txt VARCHAR(100),
        n  INT,
        PRIMARY KEY(id));
mysql> INSERT INTO mytable (txt, n) VALUES('abc', 123);
mysql> INSERT INTO mytable (txt, n) VALUES('efgsd', -4);
mysql> INSERT INTO mytable (txt, n) VALUES(NULL, 0);
mysql> SELECT * FROM mytable;
  id | txt   | n
  ---+-------+------
   1 | abc   |  123
   2 | efgsd |   -4
   3 | NULL  |    0
mysql> exit
```

25.2 Administration Tools

There are countless admin tools available for MySQL and MariaDB. The `mysql` and `mysqldump` commands and some other text-based tools are available by default. Optionally, you can install various other programs, but some of them require a graphical desktop. This section provides an overview of the most important tools. The `mysqldump` backup command, for which further information follows in Section 25.3, is not included in this overview.

25.2.1 mysql

The `mysql` program is not the database server (whose program name is `mysqld` or `mariadbd`), but a command line client. This allows you to connect to the MySQL or MariaDB server and then run SQL commands. As far as `SELECT` commands are concerned, the program displays the query results in text mode.

When starting `mysql`, you typically specify the MySQL user name with `-u`. The `-p` option allows you to enter the corresponding password after the start. This option is not available for authentication method 2 (assignment between MySQL and Linux accounts).

If the MySQL server is not running on the local computer, you need to specify the host name with `-h name`. Optionally, you can also specify a default database, which then applies by default to all subsequent SQL commands:

```
user$ mysql -u root -p mydatabase
Enter password: *******
```

Then you can interactively enter SQL commands, which must end with a semicolon. The `mysql`-specific `status` command, which can also be executed without a semicolon, displays information about the current connection as well as key data of the MySQL server. Ctrl+D terminates the program:

```
mysql> status
mysql  Ver 8.0.32 for Linux on aarch64 (Source distribution)

Connection id:        20
Current database:
Current user:        root@localhost
...

Server characterset:    utf8mb4
Db     characterset:    utf8mb4
Client characterset:    utf8mb4
Conn.  characterset:    utf8mb4
UNIX socket:        /var/lib/mysql/mysql.sock
Binary data as:        Hexadecimal
Uptime:            29 days 3 hours 59 min 28 sec
Threads: 2  Questions: 29442173  Slow queries: 0  Opens: 3419
Flush tables: 3  Open tables: 2703  Queries per second avg: 11.683
```

mysql in batch mode

For admin purposes and especially for importing backups, `mysql` can also process SQL commands from a `*.sql` file:

```
user$ mysql -u root -p mydatabase < commands.sql
Enter password: *******
```

25.2.2 mysqladmin

`mysqladmin` makes it possible to perform various admin tasks as commands. For all `mysqladmin` commands, there are also equivalent SQL commands (such as `CREATE DATABASE`). The advantage of `mysqladmin` is that it helps to automate recurring tasks by means of scripts.

As with `mysql`, you can specify the user name with the `-u` option and the host name (`localhost` by default) with `-h`. The `-p` option without any additional information prompts an interactive password query. You can avoid this by using `-ppassword` (without a space after `-p`). However, this convenience has the disadvantage that the password can be seen in plain text in the process list. An overview of the commands available for `mysqladmin` is provided by `mysqladmin --help`. The following lines give some examples. The first command creates a new database, the second one returns a list of

all MySQL status variables, and the third one returns a list of all active MySQL threads (connections):

```
user$ mysqladmin -u root -p create newdatabase
Enter password: *******
user$ mysqladmin -u root -p extended-status
...
user$ mysqladmin -u root -p processlist
...
```

25.2.3 MySQL Workbench

MySQL provides a high-quality administration program with a graphical user interface called MySQL Workbench (see Figure 25.1). You can use it to monitor the database server; change its settings; set user permissions; set up new databases; read, modify, and backup existing databases; develop database schemas; and more.

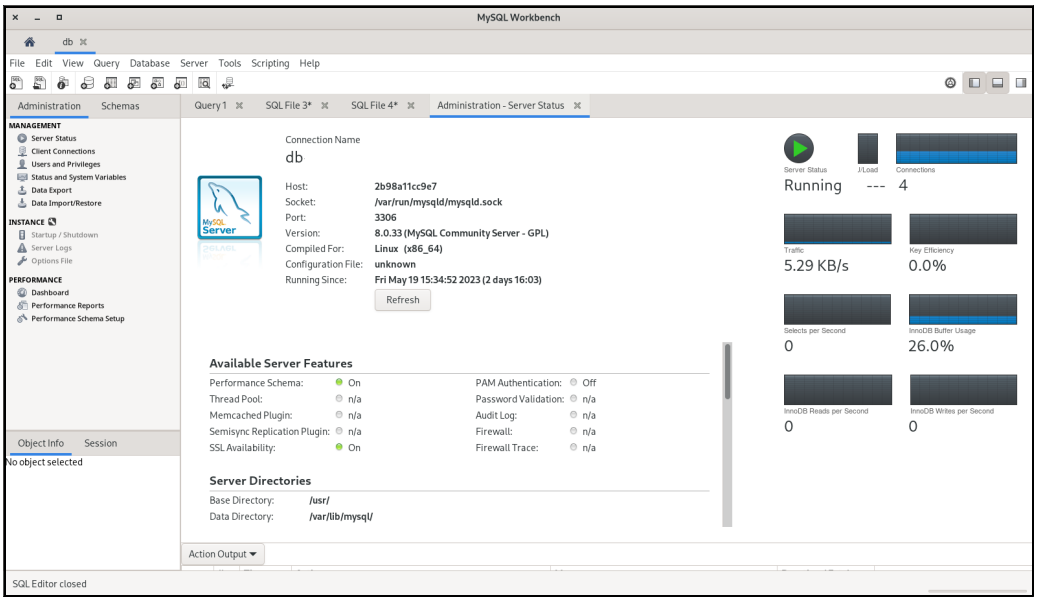


Figure 25.1 MySQL Workbench

MySQL Workbench also works in combination with a MariaDB server. When the connection is first established, MySQL Workbench warns of compatibility issues. However, these only occur when you use relatively new features where MySQL and MariaDB differ. The configuration functions for changing `my.cnf` are generally locked.

Only a few distributions provide their own packages for MySQL Workbench. Fortunately, there are suitable packages on the MySQL website that can be installed without any problems (see *https://dev.mysql.com/downloads/tools/workbench*).

25.2.4 phpMyAdmin

The most popular MySQL administration program is phpMyAdmin. Its biggest advantage is that you can operate the program through a web browser. The actual phpMyAdmin code runs directly on the web server. That is why phpMyAdmin works even if the MySQL server is configured to not allow connections over the network for security reasons. PHP then communicates with the local MySQL server via the socket file.

Many distributions contain ready-made phpMyAdmin packages:

```
root# apt install phpmyadmin       (Debian, Ubuntu)
root# systemctl reload apache2
root# dnf install phpmyadmin       (RHEL)
root# systemctl reload httpd
```

Alternatively, you can download phpMyAdmin from *https://www.phpmyadmin.net*, then install all the files into a directory that can be accessed by the Apache web server. However, if you choose this installation option, you will have to take care of updates yourself on a regular basis. You also need to make sure that the PHP extension for accessing MySQL databases is installed:

```
root# dnf install php-mysqlnd     (RHEL)
root# apt install php-mysql       (Debian/Ubuntu)
```

After the installation, you can usually access the admin interface via the following address: *https://hostname/phpMyAdmin* (see Figure 25.2).

To log into the phpMyAdmin interface, you will need a MySQL or MariaDB user name and password to match. On Debian and Ubuntu, where only the local `root` user has access to the database server by default, you must set up an appropriate user (`root2` in the example from Section 25.1).

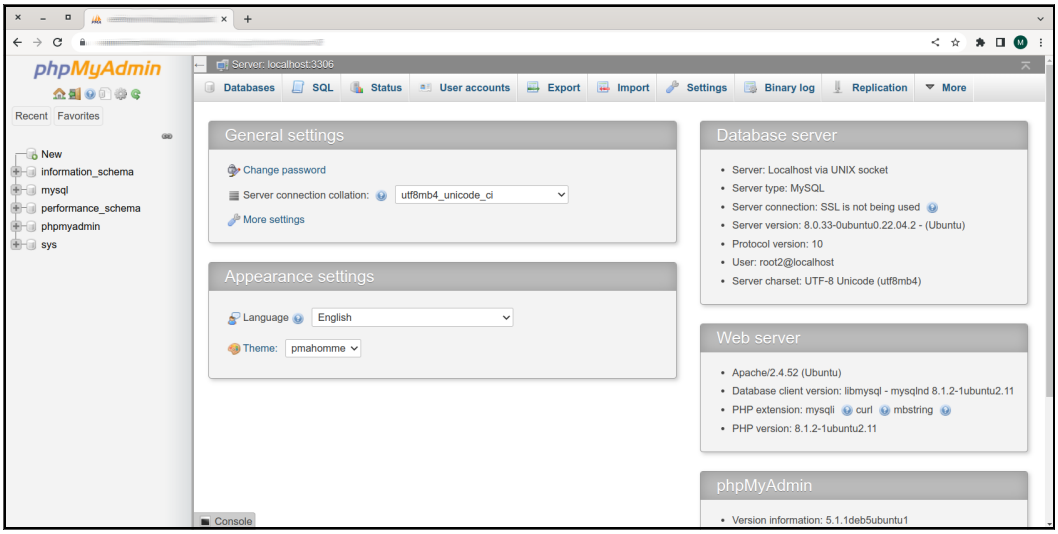


Figure 25.2 Administrating MySQL with phpMyAdmin in Web Browser

NGINX

On Ubuntu, the phpMyAdmin package does not provide for NGINX integration. You can skip the corresponding question of the setup program (where only Apache and Lighttpd are available for selection), and then set up the following symbolic link:

```
root# ln -s /usr/share/phpmyadmin /var/www/html/phpmyadmin
```

Securing phpMyAdmin

phpMyAdmin is a popular gateway for crackers. There are countless automated tools on the internet that scan web servers for poorly secured or outdated phpMyAdmin installations. A missing or easily guessable MySQL `root` password gives the attacker MySQL admin privileges! For this reason, you should take the following precautions:

- Back up the MySQL installation before installing phpMyAdmin (see Section 25.1.1).
- Assign an alias to the phpMyAdmin directory that cannot be guessed easily. *https://a-company/pMa1* is more secure than *https://a-company/phpmyadmin*, for example. Alternatively, you can of course use your own subdomain for phpMyAdmin, such as *https://myadm.a-company.com*. This requires a somewhat more elaborate Apache or NGINX configuration.
- Accept only the secure `https` protocol for the phpMyAdmin directory, not `http`.
- Secure the phpMyAdmin directory with a password at the level of the web server as well (see Chapter 24, Section 24.4).

- Make sure that your phpMyAdmin version is up to date. In some distributions, the phpMyAdmin packages are unfortunately poorly maintained. In that case, it's safer to install phpMyAdmin manually instead of through distribution packages and to update it regularly.

25.3 Backups

Even if you don't plan to specialize in database administration, you should know how to perform an error-free backup of a MySQL or MariaDB database. There are more variants and variations than you might think possible. This section introduces the `mysqldump` command. If you need continuous backups in addition to those performed at specific times, you should enable (binary) logging. This way, you can record every change to the database in a logging file. The logging files can also be used as a basis for replicating the database to a second server.

25.3.1 mysqldump

The `mysqldump` command, which is included with MySQL, creates a backup of a MySQL or MariaDB database in the form of SQL statements. The resulting file can later be imported back into an existing database using `mysql`. The basic syntax looks as follows:

```
user$ mysqldump -u root -p [options] databasename > backup.sql
```

Options

Details of the backup can be controlled by countless options (see also `mysqldump --help`). A typical backup command looks like this:

```
user$ mysqldump -u root -p --skip-opt --single-transaction  \
        --disable-keys --create-options --quick --no-tablespaces \
        --extended-insert --add-drop-table dbname > backup.sql
```

The following list explains the options used in the `mysqldump` command:

- By using `--skip-opt`, you can disable some standard options that are intended for the obsolete MyISAM table type.
- `--single-transaction` causes the entire backup to be performed as part of one transaction. This eliminates the possibility of data changing during the backup, which can lead to inconsistent links between tables.
- `--disable-keys` causes index updating to be temporarily disabled when the data is loaded later. The index is only completely regenerated at the end, which is much faster.

- Thanks to `--create-options`, `mysqldump` uses all MySQL-specific options of the `CREATE-TABLE` command in its output.
- `--quick` causes `mysqldump` to fetch the tables record by record from the server instead of reading them all at once. This is more efficient for large tables.
- `--no-tablespaces` prevents the *access denied* error in MySQL 8. Alternatively, you can assign the process privilege to the user who performs backups using `mysqldump`.
- Because of `--extended-insert`, `mysqldump` generates `INSERT` commands that insert multiple records at once, which not only reduces the size of the backup file a little, but also speeds up restoring the data later.
- `--add-drop-table` causes `mysqldump` to prefix a `DROP-TABLE` command to each `CREATE-TABLE` command. This avoids errors if individual tables already exist in the database—for example, due to an incompletely imported backup.
- If you want to back up *all* databases (not only a specific database), you must specify the `--all-databases` option.

Restoring the database

To restore a database from a backup, you must first create the respective database (if it does not already exist). Then pass the backup file to `mysql`. The `--default-character-set` option makes sure that the character strings stored in UTF8 format are actually read in this format:

```
user$ mysqladmin create dbname
user$ mysql -u root -p --default-character-set=utf8 \
        dbname < backup.sql
```

Compressed backups

Backup files created using `mysqldump` are very large due to the text format. The easiest way to avoid this problem is to compress the backups immediately or decompress them directly when restoring. The corresponding commands look as follows:

```
user$ mysqldump [options] dbname | gzip -c > backup.sql.gz
user$ gunzip -c backup.sql.gz | mysql [options] dbname
```

However, the compression using `gzip` consumes a lot of CPU resources. If you want to save compute time and can accept somewhat larger backup files in return, we recommend using the `lzop` compression command from the package of the same name:

```
user$ mysqldump [options] dbname | lzop -c > backup.sql.lzo
user$ lzop -c -d backup.sql.lzo | mysql [options] dbname
```

25.3.2 Backup Tools and Variants

The `mysqldump` command works great for small databases. However, the running database operation becomes increasingly limited as the database size increases. This is a problem for database systems that need to run 24 hours a day without interruption.

Hot backup

The ideal solution to this problem would be a hot backup procedure that works trouble-free during operation. MySQL offers a corresponding backup program, but this is only available as part of a paid MySQL Enterprise license. A good alternative is the open-source program XtraBackup. However, this program is not compatible with MariaDB (see *https://www.percona.com/software/mysql-database/percona-xtrabackup*).

LVM backup

Another possibility is offered by the `mylvmbackup` command. It minimizes the time for which the database is blocked to a few seconds. However, it requires that all database files be located in a logical volume. The script creates a snapshot of this logical volume and uses it to transfer all database files to a compressed backup file. But the `mylvmbackup` Perl script was last updated in 2014 (see *https://launchpad.net/mylvmbackup*).

The backup created in this way differs from `mysqldump` results in two respects: First, it basically captures *all* databases including all additional data such as stored procedures, triggers, access rights, and so on. And second, it's in binary format. This means that only all databases can be restored together, and that a MySQL or MariaDB server with the same configuration and, if possible, the same version number is required to restore the backup.

Incremental backups through binary logging

You can use a script to automate backups with `mysqldump` or other tools to create a daily or weekly backup. To further minimize potential data loss in the event of a disaster, you can also enable incremental backups. To do this, add the following line to `/etc/mysql/my.cnf` and then restart the database server:

```
# change in /etc/mysql/my.cnf
log_bin = /var/log/mysql/mysql-bin.log
```

The database server then logs all SQL commands that modify data. The logging files are automatically numbered (`mysql-bin.000001`, `.000002`, etc.). As it is always only the last file that changes, it's relatively easy to transfer these files to a backup directory at short intervals (e.g., using `rsync`).

If you need to restore your databases, you should first perform the restore steps described previously for the last full backup. Then use `mysqlbinlog` to import any changes that have occurred since then. The command sequence looks as follows:

```
root# mysqlbinlog --start-position=<p> mysql-bin.<n> | mysql \
        -u root -p
root# mysqlbinlog mysql-bin.<n+1> | mysql -u root -p
root# mysqlbinlog mysql-bin.<n+2> | mysql -u root -p
..
```

Replication

Once binary logging is enabled, it's only a small step to replication. This allows you to synchronize a second MySQL server with the first one, so you always have an active second database system that can replace the primary system in an emergency. But replication does not replace backups because commands like `DROP TABLE` are, of course, synchronized immediately! An introduction to MySQL database replication is provided in the MySQL manual (see *https://dev.mysql.com/doc/refman/8.0/en/replication.html*).

25.4 Installing WordPress

Now that I have explained the entire setup of a LAMP/LEMP system in this and the previous chapter, I want to show you a short example of the installation of a typical web application. Due to its popularity, I chose the WordPress content management system. As you will soon realize, this section is not about using WordPress, but only about installing it. The procedure is exemplary for many other web applications.

Installing the WordPress files

For the sake of simplicity, I assume here that WordPress is supposed to be the primary application of the website. The WordPress files are therefore installed directly into the root directory of the web server. If you manage multiple apps or multiple virtual hosts on your web server, it makes more sense to provide a separate installation directory and an Apache or NGINX configuration file for WordPress:

```
root# cd /var/www/html
```

You can download the latest version of WordPress by using `wget`. `tar` unpacks the archive (this creates the `wordpress` subdirectory), while `rm` deletes the previously downloaded archive:

```
root# rm testfiles       (if necessary, remove test files)
root# wget https://wordpress.org/latest.tar.gz
root# tar xzf latest.tar.gz
root# rm latest.tar.gz
```

Setting access rights

In the next step, the access rights to the files must be set in such a way that the web server is allowed to read and edit them. A change is required so that WordPress can perform updates and save uploaded images/files.

The procedure depends on the web server and the distribution. If you use Debian or Ubuntu, the following command will suffice:

```
root# chown -R www-data.www-data wordpress    (Debian/Ubuntu)
```

On RHEL, you must also set the SELinux context so that Apache has write permission to the directory:

```
root# chown -R apache.apache wordpress        (RHEL)
root# chcon -R system_u:object_r:httpd_sys_content_rw_t:s0 wordpress
```

Apache configuration

For Apache to consider the WordPress installation directory as the root directory, you must change the `DocumentRoot` setting; on Debian/Ubuntu, this is usually in `default-ssl.conf`, and on RHEL in `httpd.conf`:

```
# file /etc/apache2/sites-available/default-ssl.conf  (Debian/Ubuntu)
# file /etc/httpd/conf/httpd.conf                     (RHEL)
<VirtualHost _default_:443>
    ServerName a-company.com
    DocumentRoot /var/www/html/wordpress
    ...
```

For the change to take effect, you must prompt Apache to reread the configuration:

```
root# systemctl reload apache2
```

NGINX configuration

If you use NGINX instead of Apache, you must change the `root` parameter in its configuration:

```
# file /etc/nginx/sites-available/default (Ubuntu)
server {
  server_name lemp.a-company.com;
  root /var/www/html/wordpress;
  ...
}
```

The changes will only take effect after a reload:

```
root# systemctl reload nginx
```

Setting up the database

WordPress requires access to a MySQL or MariaDB database. You can use the following commands to set up the `wp` database and the `wpuser` database account:

```
root# mysql
MariaDB/MySQL> CREATE DATABASE wp;
MariaDB/MySQL> CREATE USER wpuser@localhost IDENTIFIED BY 'secret';
MariaDB/MySQL> GRANT ALL ON wp.* TO wpuser@localhost;
```

Starting the WordPress configuration

This completes all the preparatory work. In a web browser, you can now visit *https://a-company.com/wp-admin/setup-config.php*, replacing *a-company.com* with the name of your own domain as usual. If everything worked, you will see the WordPress configuration home page in the browser. The **Let's Go!** button takes you to a form where you need to enter the connection data for the database (see Figure 25.3).

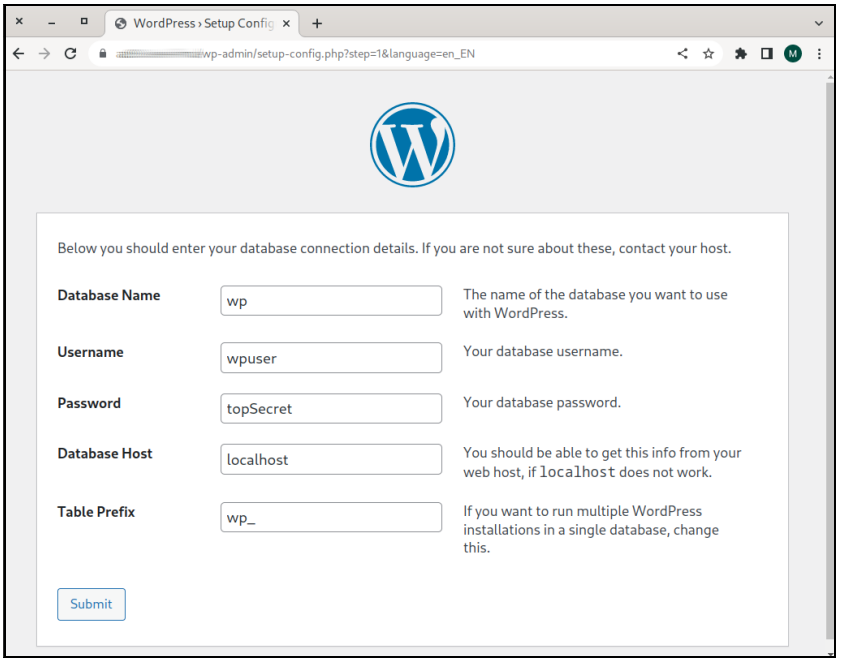


Figure 25.3 WordPress Database Configuration

If WordPress can connect to the database and write the appropriate `wp-config.php` configuration file, you will be taken to another dialog. There you must enter the key data of your WordPress installation—that is, the desired name of your website and the user name for further WordPress administration. For security reasons, do not use `root`, `admin`, or `wp-admin`, and also do not use the MySQL account name.

In the WordPress administration interface (*https://a-company.com/wp-admin*), you can now set the layout of your web presence, write your first blog posts, and so on. However, none of this is the subject of this section.

Backups As soon as WordPress is basically running, you should think about a backup system. Should you ever find yourself in the situation where you need to restore WordPress from the backups, you will need both the database backup (which contains all the posts, pages, and comments on your website) and the contents of the `wordpress` directory (which contains WordPress itself as well as all installed plugins and all downloaded images and other media files).

This means you need to back up both the contents of the MySQL/MariaDB database `wp` and the entire `/var/www/html/wordpress` directory on a regular basis. For suggestions on how you can perform such backups in practice, see Chapter 28.

Chapter 26
Postfix and Dovecot

This chapter describes how you can set up an email server. This type of server allows you to provide all employees of a company or organization with their own email addresses. Every employee can send emails via SMTP and read, save, and manage them via IMAP. The Postfix program is used as the SMTP server, while the Dovecot program is used as the IMAP server and for the SMTP authentication. These programs are available in all common distributions.

You can try to control the flood of spam by using SpamAssassin. (I'll tell you this much in advance: SpamAssassin doesn't work miracles either.) If your employees or the employees of your clients work on Windows PCs, it may also be worth installing the ClamAV virus protection program. Before you can get to work, you need a server with an internationally valid host name. You must be able to configure its DNS entries yourself (MX entry, reverse DNS).

Email is a much more complex topic than many newcomers to the subject assume. Different programs and commands are available for each subtask, and there are an almost infinite number of configuration options. That is why this chapter first describes the basic principles of email communication and then provides an initial overview of the tools available to you.

26.1 Introduction and Basic Principles

Email is pretty simple, right? From the end user's point of view, this is true—at least as long as everything works. But behind the scenes, the email system is much more complex than it appears. There are many configuration options, all of which are justified under certain circumstances. Good books on email servers such as Sendmail, Postfix, or Exim often contain more than 1,000 pages! So it's obvious that I can only go into the basic configuration here.

26.1.1 Components of an Email Server

A complete email server consists of three components:

- **Mail transfer agent**
  The *mail transfer agent* (MTA) is what is colloquially referred to as an

email server. The MTA takes care of sending or receiving emails through the internet, using the SMTP protocol.

Most newcomers to the inner workings of the email world do not realize that the MTA's responsibility is limited to network traffic and ends at the server. Received emails are then forwarded to the MDA, which takes care of the local storage. It is *not* the MTA's job to deliver emails to a user who is usually working on a different computer!

Examples include Courier, Cyrus, Exim, Postfix, Qmail, and Sendmail.

- **Mail delivery agent**
  The *mail delivery agent* (MDA) is responsible for the local delivery of emails—that is, the storage of emails arriving at the MTA in local mailboxes. A *mailbox* in this context is simply a directory or file on the server.

  Examples include Maildrop and Procmail.

  Some MTAs have an integrated MDA or one is supplied with them, such as the `local` command for Postfix. Nevertheless, the Maildrop and Procmail programs are popular because they offer even more configuration options.

- **POP and/or IMAP server**
  Emails are rarely read directly on the server. The POP and IMAP protocols have become established so that an external user can transfer emails to their local computer or smartphone or manage them from there. To support these protocols, a POP and/or IMAP server must be set up on the server. POP and IMAP servers are sometimes counted as MDAs, but this contradicts the original meaning of an MDA and is therefore incorrect.

  Examples include Courier-IMAP and Dovecot.

In this context, you will often come across a number of acronyms: Email programs (clients), as the user sees and uses them, are called *mail user agents* (MUAs) in the terminology of the email world. These programs retrieve emails from the email server (POP protocol) or help with reading and managing external emails (IMAP). For sending, the MUA communicates directly with the MTA (SMTP protocol). Popular representatives of this genre include Thunderbird, Evolution, KMail, Microsoft Outlook, Apple Mail, and the text-based Mutt program. Instead of a local mail client, web applications are increasingly being used, such as Google Mail.

To help you keep track, Table 26.1 summarizes the most important abbreviations from the email environment. Some abbreviations have not yet appeared in the text but will appear in the next pages.

Abbreviation	Meaning
IMAP	Internet Message Access Protocol
MDA	Mail delivery agent
MTA	Mail transfer agent
MUA	Mail user agent
POP	Post Office Protocol
SASL	Simple Authentication and Security Layer
SMTP	Simple Mail Transfer Protocol

Table 26.1 Important Email Abbreviations

Email Then and Now

In the early days of the internet, *every* computer was directly connected to the internet and had its own email server if required. POP or IMAP were not necessary because the users received the emails directly on their computer (which functioned as a server). The standard text-based mail clients (`elm`, `mail`, `pine`) at that time were able to read the local mailboxes directly—a capability that most modern mail clients with a graphical user interface have lost. Mail clients at that time also did not communicate with the MTA via SMTP, but simply passed the email to be sent to the `sendmail` command.

Exchange Server

Microsoft has its own way of handling emails by using Exchange Server. By default, Exchange Server uses its own protocols to communicate with the email client, which is why most users will use Outlook, a web interface, or an Exchange Server–compatible app. POP, IMAP, and SMTP are only supported with special configuration or with additional software. The only Linux mail client that works reasonably well with Exchange Server is Evolution.

26.1.2 Protocols and Ports

POP, SMTP, and IMAP are different protocols for transferring emails:

- **SMTP**
  The *Simple Mail Transfer Protocol* is used to send emails. Local programs can typically communicate with an SMTP server running on the same computer without authentication. If, however, an external mail client

sends an email, it transfers the message to the SMTP server and must authenticate itself in the process.

- **POP**
  In the past, the *Post Office Protocol* was used to transfer emails from the mail server to a local computer (client). This protocol is perfect if users only use *one* email program and then save the downloaded emails on their own computers. However, POP is not suitable if several mail clients are used at the same time.
- **IMAP**
  The *Internet Message Access Protocol* is an alternative to POP. The main difference from POP is that with IMAP the emails usually remain on the IMAP server and are organized there in several directories (mailboxes). In this case, the email client is only used to communicate with the server. IMAP is ideal if you want to process your emails from different computers, smartphones, tablets, and so on.

Ports

Table 26.2 summarizes which ports are provided for the various mail protocols.

Port	Usage
25	Unencrypted/encrypted SMTP, communication between two servers
110	Unencrypted POP
143	IMAP with STARTTLS or unencrypted
465	Encrypted SMTPS (deprecated)
587	SMTP with STARTTLS specially designed for mail clients
993	Encrypted IMAP
995	Encrypted POP

Table 26.2 Ports for Email Communication

Port 587 is particularly noteworthy: mail clients are supposed to communicate with the SMTP server via this port, while the communication between two mail servers usually takes place through port 25. Port 25 was originally designed for nonencrypted communication, but it's now also frequently used with encryption.

Mailbox formats

The mail server stores incoming emails in local files or directories. The *mbox* format is often used for this purpose: all emails from an account are simply combined into one long text file. The usual filename is `/var/spool/mail/accountname`. Lines beginning with `From` are used to separate the

emails. The format is documented on the internet—for example, at *https://www.commandlinux.com/man-page/man5/mbox.5.html*.

An alternative to the mbox format is the *maildir* format, where each individual email is stored in its own file, often in a directory within the user's home directory. A mailbox consists of all files within a directory. The obvious advantage is that individual messages can be deleted more easily. The maildir format is particularly recommended if the mail server is combined with an IMAP server.

Reading local emails

Traditionally, Linux computers use email as a local communication medium. Many distributions have Postfix installed by default. Emails with error or status messages are sent to `root@localhost` and then end up in `root`'s mailbox, often in the `/var/spool/mail/root` file. There is a high risk that you will never see such emails.

The easiest way to read such messages is to use the Mutt program (see Chapter 11, Section 11.5). If you install this program and run it in a terminal window as `root`, you can read the local mails.

A more elegant solution is to use `/etc/aliases` to redirect all emails addressed to `root` to the inbox of the user who is normally responsible for the administration of the computer:

```
# at the end of /etc/aliases
...
root:   kofler
```

The changed setting will not take effect until you run the `newaliases` command. However, you still need a mail client that analyzes the local mailbox. Most graphical mail clients like Thunderbird, KMail, or Evolution can be configured accordingly.

26.1.3 The Message Flow in Detail

You can use Figure 26.1 to track how an email is sent from Mr. Hoover (`hoover@a-company.com`) to Ms. Smith (`smith@target.com`) or how an email is sent from `anyone@sender.com` to `hoover@a-company.com`.

Let's start with the first case: Mr. Hoover, an employee of `a-company.com`, is sitting in front of his notebook computer at an airport, composes an email in the Thunderbird email client to `smith@target.com`, and sends it off. Thunderbird now contacts the company's mail server via the SMTP protocol, via the Postfix program running on the `a-company.com` server. Postfix takes the email, determines which domain it is addressed to, and in turn contacts the mail server of target.com. The MTA of target.com verifies that the specified

user smith actually has an email account on target.com and accepts the email.

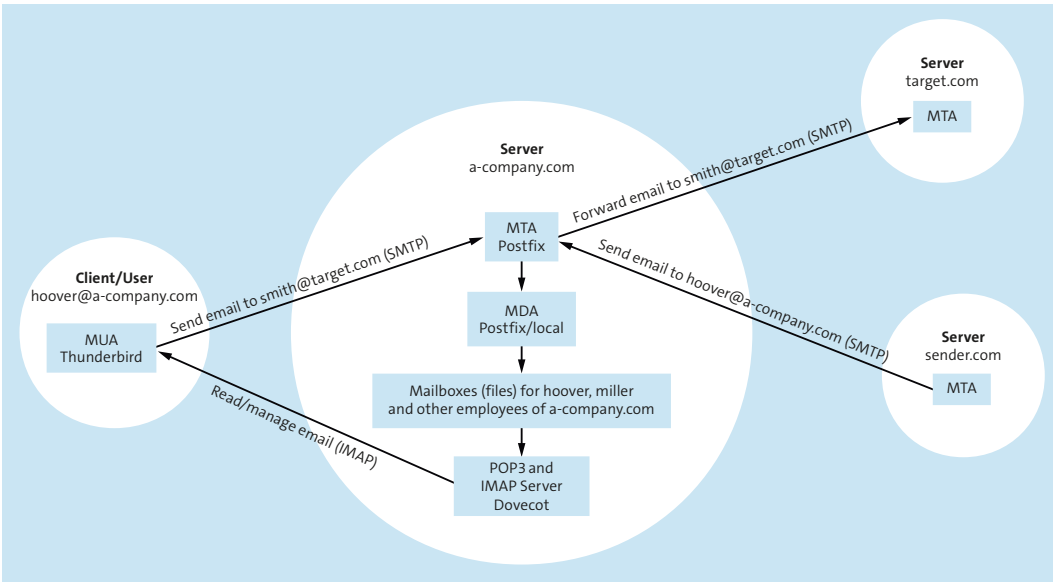


Figure 26.1 Email Communication Flow

Meanwhile, `anyone@sender.com` wrote an email for Mr. Hoover and sent it. The MTA of `sender.com` now contacts the MTA of `a-company.com` and passes the email on. On the `a-company.com` server, Postfix detects that there is an email account for hoover. Postfix accepts the email and passes it to the local command belonging to Postfix, which stores the email in Mr. Hoover's mailbox. The mailbox is a file or directory located on the `a-company.com` server.

The email remains there until Mr. Hoover's email client contacts the Dovecot program on the `a-company.com` computer via the POP or IMAP protocol. Most email programs are set to do this automatically every few minutes. Of course, if Mr. Hoover is offline at the time, it may take hours or days to get there. Now the email only needs to be transferred to the email program and can then be read there.

Host Names

For the sake of simplicity, only the domain names of the companies are mentioned in the previous paragraphs and in Figure 26.1. As a matter of fact, mail servers and clients obviously use full host names. These can vary depending on the protocol and can be, for example, `imap.a-company.com`, `mail.destination.com` or `a-hostname.sender.com`.

26.1.4 Variants and Options

In order not to overwhelm you with details, I have shortened the previous description of an email communication a little. For example, I did not address how to deal with temporarily undeliverable emails. An email server makes delivery attempts for several hours before it gives up sending. If the delivery definitely fails, the sender receives the email back with a brief description of the cause of the error.

Authentication

The topic of authentication has also been left out. Not everyone is allowed to just contact any MTA and send emails. To prevent such uncontrolled email sending, emails may only be sent locally in the basic configuration. Authentication is required so that an external email client can also send emails. Although the Postfix program presented in this chapter supports the Simple Authentication and Security Layer (SASL) protocol, it cannot perform the authentication itself. In the sample configuration in this chapter, the Dovecot program performs this task.

Encryption and STARTTLS

To prevent the content of emails from being transmitted in plain text, the connection between the client and the mail server or the connection between two mail servers (MTAs) must be encrypted. Most mail servers support the Transport Layer Security (TLS; formerly SSL) protocol for this purpose. Encryption is usually initiated with STARTTLS when the connection is established. The communication begins unencrypted; the client and server then agree on the best possible encryption method and continue the communication in encrypted form. This works great with many mail clients and servers.

The encryption outlined here only concerns the *transmission* of mail from one server to another. It has nothing to do with whether the *content* of the message itself is encrypted or at least signed. The mail client is responsible for encrypting or signing the content for the sender and displaying the message correctly for the recipient. Despite the undisputed advantages of signed or encrypted emails, you will rarely come across such emails in real life. Convenience, the relatively complex key management, and two incompatible standards (PGP and S/MIME) stand in the way of widespread use. This makes it all the more important that the email, if it is not encrypted itself, is at least transported in a tap-proof manner.

Relaying

Another point is the message transfer from one MTA to the next. The path is not always as direct as in Figure 26.1. Sometimes the transfer takes place via several stations. The intermediate MTAs only pass on the message (relaying). This is especially useful if there is one primary server and one or more backup servers for a mail domain. If the primary server is not available at a given moment, the backup servers receive the emails and forward

them to the primary server later. This type of configuration mitigates the risk of the email system being unavailable during maintenance.

Make Sure Your Mail Server Is Secure Lest It End Up on a Blacklist!

The worst thing that can happen when you configure an email server is that the relaying process is inadequately or incorrectly secured: then anyone can send messages to your email server for forwarding without authentication.

Spammers constantly search the internet for such servers and abuse them for the omnipresent flood of advertising. This entails unpleasant consequences: within a few days, your server will end up on blacklists that list dangerous or misconfigured email servers. Many email servers do not accept emails from such servers to avoid spam. Unfortunately, it's much more difficult to be deleted from such blacklists than to end up on them.

So be careful with the configuration! If you suspect that your server (more precisely, its IP address) has ended up on a blacklist, you can verify that very easily on the following page: *https://mxtoolbox.com/blacklists.aspx*.

Spam and virus protection, web interface

The scenario shown in Figure 26.1 can be further extended with server-side spam and virus protection. If you want your users to be able to read emails directly on a web page without a mail client, you also need a web interface, such as the RoundCube program or Horde.

26.1.5 DNS Configuration

Let me return again to Figure 26.1: How does the MTA on the `a-company.com` computer know the IP address of the mail server for `target.com`?

Because of the Domain Name System (DNS) entries, of course, you will answer.

MX entry

This is basically correct, but it's not ordinary DNS entries (so-called A records) that are responsible for email traffic, but special MX entries. An MX entry specifies the host name, not the IP address, of the computer responsible for a domain's email. This makes it possible to implement the email services on a different machine than the rest of the internet services such as web, SSH, FTP, and so on. If your mail server also supports IPv6, you need an AAAA record with the IPv6 address of the mail host. The MX entry does not change.

In addition, each MX entry contains a priority number. If multiple email servers are set up with different priorities, the server with the highest priority will typically receive all emails. If this server is temporarily unavailable, the lower-priority servers come into play. These servers usually only serve

as a backup system and forward the emails to the primary server (relaying) as soon as it's online again.

Table 26.3 summarizes a typical DNS configuration for a simple server. All Internet services, including email, run on the same computer; there is no backup email server. As the MX entry contains a domain name (not an IP address), the domain name specified there must also be defined by an A record. In the sample configuration, the full host name of the computer is `efd.a-company.com`. The MX entry points to this host name.

Type	Name	Value	Priority
A	a-company.com	95.217.156.167	
A	efd.a-company.com	95.217.156.167	
A	www.a-company.com	95.217.156.167	
AAAA	a-company.com	2a01:4f9:c012:7e31::1	
AAAA	efd.a-company.com	2a01:4f9:c012:7e31::1	
AAAA	www.a-company.com	2a01:4f9:c012:7e31::1	
CNAME	imap	efd	
CNAME	mail	efd	
CNAME	pop	efd	
CNAME	smtp	efd	
MX	–	efd.a-company.com	10
TXT	a-company.com	"v=spf1 "…	

Table 26.3 Sample DNS Configuration of Simple Web and Mail Server with IPv4 and IPv6

CNAME entries

The `CNAME` entries are aliases, so to speak. They do not refer to IP addresses, but to other host names, and therefore work simultaneously for IPv4 and IPv6. The `CNAME` entries for `imap`, `pop`, and `smtp` listed in the example are not required for mail server operation, but often facilitate the client configuration. Some email clients assume by default that there are host names beginning with `smtp`, `imap`, and possibly also `pop` or `pop3` for the corresponding protocols.

TXT entries

The `TXT` entry contains information about the SPF spam protection system, which is described on the following pages. In addition to the entries specified in Table 26.3, each domain usually has other DNS entries that point, for example, to additional hosts or to the underlying name servers.

Finally, there is the question of how or where you carry out the DNS configuration. Usually you use a web interface provided by the service provider with whom you have registered your domain name (see Figure 26.2).

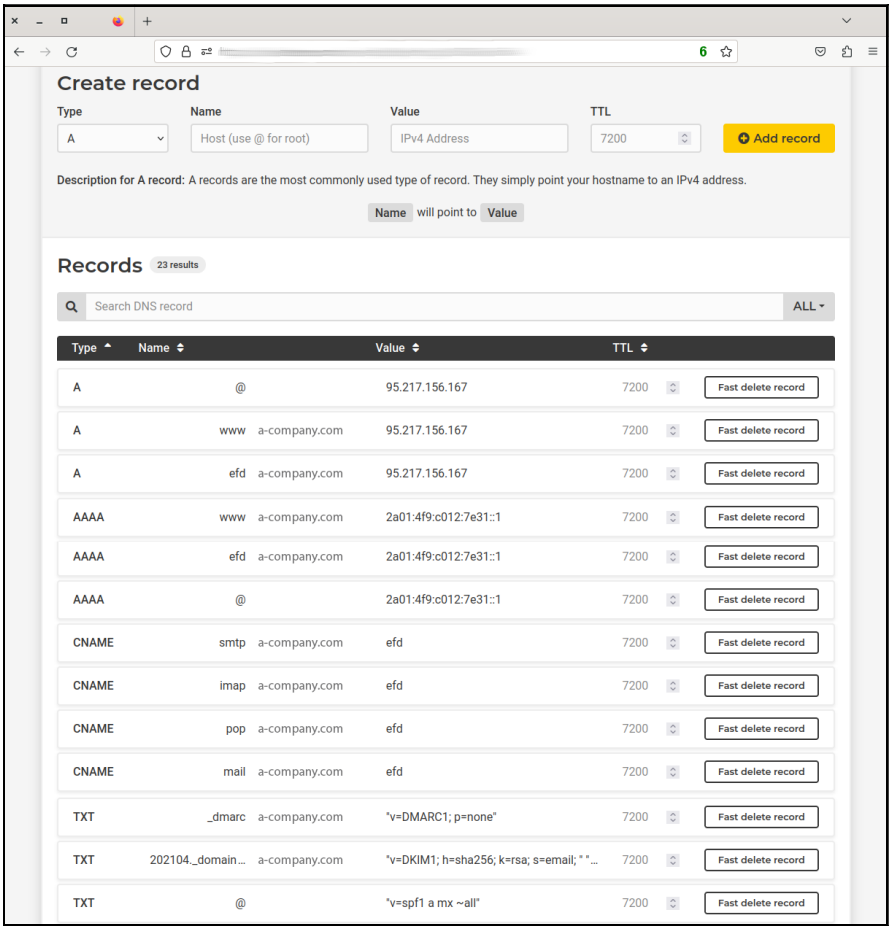


Figure 26.2 DNS Configuration with Domain Name Registrar

DNS configuration
host command

DNS entries are public. You can use the `host` command to read the entries for a specific host name. The following lines show how you can first determine the host name(s) of the mail server(s) and then query their IP address:

```
user$ host -t MX a-company.com
a-company.com mail is handled by 10 efd.a-company.com.
user$ host efd.a-company.com
efd.a-company.com has address 95.217.156.167
efd.a-company.com has IPv6 address 2a01:4f9:c012:7e31::1
```

Do not run `host` directly on the root server, but on an external computer! Otherwise, your own name server as well as the name server of your

provider may influence the result. This makes it difficult to search for any configuration errors of your own.

26.1.6 Reverse DNS Entry

Usually, DNS servers provides the IP address associated with a host name. If an external computer wants to contact `a-company.com`—be it with a web browser, via SSH or by email—it first contacts the next DNS server. This returns the IP number of the server of `a-company.com`. For mail traffic, the resolution is done in two stages: first the host name of the mail server is determined via the MX entry, then its IP address. *Reverse DNS* works the other way around: the IP number is known, but the host name is now searched for. For this to work, a reverse DNS entry must be present. If you are renting a root server, your provider must perform this reverse DNS entry. Many providers make a corresponding configuration tool available to their customers.

The reverse DNS entry has nothing to do with the other DNS entries! This is why the setting is not made on the DNS configuration page of your domain name, but as part of the configuration of your root server or your cloud instance.

The easiest way to test reverse DNS is to use the `host` command. Note that although several host names can lead to the same IP address (e.g., if several websites are running on one server thanks to *virtual hosting*), the reverse IP resolution always provides only one unique host name:

```
user$ host -t MX a-company.com
a-company.com mail is handled by 10 efd.a-company.com.
user$ host efd.a-company.com
efd.a-company.com has address 95.217.156.167
efd.a-company.com has IPv6 address 2a01:4f9:c012:7e31::1
user$ host 95.217.156.167
167.156.217.95.in-addr.arpa domain name pointer efd.a-company.com.
user$ host 2a01:4f9:c012:7e31::1
1.0.0.0.0.0...0.a.2.ip6.arpa domain name pointer efd.a-company.com.
```

Reverse DNS entries are not actually mandatory for internet traffic. There is no internet standard that requires such entries. Nevertheless, Reverse DNS entries have become established for email servers. This is because many MTAs only accept the receipt of emails from external MTAs if a reverse DNS entry exists for the host name of the server on which the MTA is running, and this is presumably not dynamically generated. This protective measure is intended to combat computers or IoT devices infected by malware, which are misused to send spam—often without the owner's knowledge.

Even though the reverse DNS test does not provide reliable protection against spam, it is often used. This means that to prevent other MTAs from classifying mails from your server as spam, the server needs a reverse DNS entry and, if necessary, a second one for the IPv6 address.

SPF and DKIM

Basically, the DNS configuration is now sufficient to set up and test a mail server. If everything works, you should also carry out Sender Policy Framework (SPF) and DomainKeys Identified Mail (DKIM) configurations. Both methods help you to check whether a mail has been sent from the correct server.

The practical benefits of the two methods are not undisputed. However, they are still worth implementing because they increase the chances that emails sent from your server will not be classified as spam by the recipient. Details on this topic follow in Section 26.9.

26.2 Postfix (MTA)

This section describes how you can install and configure the Postfix MTA on RHEL and Ubuntu. Postfix is currently one of the most popular MTAs and is very well documented—both on *https://www.postfix.org* and in countless articles and some books.

Nevertheless, the choice of Postfix is by no means a foregone conclusion. While there is only one widely used program for some tasks, the open-source world offers several excellent MTAs to choose from. Whether Postfix or Exim, Qmail or Sendmail—all these programs are widely used and fulfill their purpose. If you talk to administrators, everyone will probably recommend the MTA they use themselves and know well.

Prerequisites

Before you follow me through the various configuration details on the following pages, a few prerequisites must be met:

- You need an email account independent of your server (e.g., at GMX, Google, or mailbox.org) to test your new email system.
- The DNS configuration of your domain must be correct.
- The host name of your computer must have been set correctly. The `hostname` and `cat /etc/hostname` commands should each return the correct name of your server. In the example that runs through the entire chapter, this hostname is `efd.a-company.com`.
- There should be a reverse DNS entry for your root server.
- Finally, the installation of a small text-based email client is recommended to test the MTA directly (see Chapter 11, Section 11.5).

26.2.1 Installation on Debian and Ubuntu

On Debian and Ubuntu, a configuration program appears after the installation of the `postfix` package. There you need to specify what kind of basic installation you want. On a root server, **Internet Site** is the right choice: you want to use Postfix to send and receive emails on the server via SMTP.

In the next step, you need to specify the domain for the email server—for example, `a-company.com`. This name is used to complete email addresses without domain names. So an email to `name` becomes an email to `name@a-company.com`.

Basic configuration

After this tiny configuration, Postfix starts immediately. The program performs the following functions in the basic configuration:

- Postfix receives emails via SMTP to `name@a-company.com`. If there is a `name` login on the server, the email will be accepted and saved. This applies to all accounts defined in `/etc/passwd`. For example, if Apache is installed, `www-data@a-company.com` is also a valid email address. If `name` is unknown, the email will be rejected (*user unknown*).
- Accepted emails are stored by Postfix's own MDA in mbox format in the `/var/mail/name` file. The files in `/var/mail` are therefore the mailboxes of the various email users of the computer. In simple terms, the mbox format means that the emails are joined together in a file that grows larger and larger. Because emails are usually encoded in a text format, you can even view the mbox file by using `cat` or `less` if necessary. An alternative to the mbox format is the maildir format, in which there is a separate directory for each mailbox and a separate file for each email.
- Local users can send emails—both internally to all accounts on the server and externally to any other email addresses.

Test

As a first test, you can send an email from an external account to `name@a-company.com`, where `name` is an active Linux account on the server. The email should show up in `/var/mail/name` after a short time. If you log in as `name`, you can read the email using Mutt. Next, use Mutt again to test sending an email to your external email address. If problems occur during the two tests, the most likely cause of the error is an incorrect or missing DNS configuration.

26.2.2 Installation on EHEL

On RHEL, Postfix is usually already installed and active. You can check this for yourself by using `systemctl status`. If Postfix is not yet running with a minimal installation, you should install and activate the program as follows:

```
root# dnf install postfix
root# systemctl enable --now postfix
```

Basic configuration

Unlike Debian and Ubuntu, you do not have a choice of predefined configuration variants. In the default configuration, Postfix can only send email from local users to the outside world and receive email from local users as well. However, the mail server cannot accept emails from outside.

You must explicitly enable all other functions by making the appropriate settings in `/etc/postfix/main.cf`. In doing so, you can use the listing in Section 26.2.4. In particular, `main.cf` must contain the `inet_interfaces=all` setting. You also need to specify matching names for `myorigin` and `myhostname`. Note that the change of `inet_interfaces` doesn't take effect until you restart Postfix!

In the default configuration, `main.cf` is quite confusing due to countless comments. To address this, create a backup of the file and then a cleaned up version for further editing:

```
root# cd /etc/postfix
root# cp main.cf main.cf.orig
root# grep -Ev '^#' main.cf.orig > main.cf
```

Firewall

To ensure that the mail server can be reached from the internet, ports 25 and 587 must not be blocked by a firewall. This is exactly the case with Fedora, RHEL, and SUSE. Although this does not matter for the default configuration of Postfix, if you want your mail server to be accessible from outside, you must open some ports of the firewall.

On RHEL, you first need to determine which firewall zone is assigned to the network interface to the outside (here the `enp1s0` interface and the `public` zone). Then open the ports using `firewall-cmd`. The command for port 465 is optional. Port 465 is for the SMTPS protocol, which is now considered obsolete:

```
root# firewall-cmd --get-active-zones   (determine active zone)
public
  interfaces: enp1s0
root# firewall-cmd --permanent --zone=public --add-service=smtp
root# firewall-cmd --permanent --zone=public \
  --add-port=465/tcp   (optional)
root# firewall-cmd --permanent --zone=public --add-port=587/tcp
root# firewall-cmd --reload
```

Chapter 29 contains background information and descriptions of other strategies for configuring a firewall. If you run your mail server in the cloud,

you need to make sure that the previously mentioned ports are not blocked by the cloud firewall. In AWS EC2, you must define appropriate exception rules in the instance's security group (often in *launch-wizard-nnn*).

Mutt

Before you run tests using Mutt, you should add the `set hidden_host=yes` line to `/etc/Muttrc`.

26.2.3 Configuration

The basic Postfix configuration files are located in `/etc/postfix`. Within the configuration files, you can use preconfigured options like variables—such as `option1 = value1` and then `option2 = $option1`. Statements in the configuration file may extend across multiple lines, but the text must be indented from the second line on.

Lookup tables

The configuration files often reference tables or lists, which are called *lookup tables* or *mappings* in the documentation. In this context, `option=type:name` syntax applies. The most common file type is `hash`. In this case, Postfix analyzes the `name.db` file; that is, it adds the `.db` extension to the specified filename. `*.db` files are tables in a binary format (Berkeley DB [BDB] format). To manipulate such files, you can use the `postmap` command.

Tables in text format are not provided for reasons of efficiency. The only exception is the `/etc/aliases` text file, whose format is compatible with Sendmail. But even in this case Postfix falls back on the associated BDB file, `aliases.db`. That is why `aliases.db` must be synchronized by using the `newaliases` command after each change to `aliases`. To learn what mail aliases are and how to configure them, see Section 26.4.

External databases

However, Postfix can also communicate with external databases (MySQL, PostgreSQL, or LDAP), provided the corresponding Postfix extension packages are installed. The file named in the configuration file now doesn't contain the actual data, but the connection information and a (SQL) query. Using external databases is especially useful if you need to manage a large number of email accounts, such as hundreds or thousands. I won't discuss this case any further in this chapter.

```
virtual_mailbox_domains=mysql:/etc/postfix/mysql-virt-domains.cf
```

26.2.4 main.cf

The main configuration file for Postfix is `/etc/postfix/main.cf`. The following listing shows the most important lines of this file in the default setting (**Internet Site** type) common on Ubuntu. As always in this chapter, the full host name of the mail server is `efd.a-company.com` and the domain name is `a-company.com`:

```
# file /etc/postfix/main.cf (excerpts)
# this is how Postfix reports to other MTAs
smtpd_banner = $myhostname ESMTP $mail_name (Ubuntu)

# no notification of new mail via 'biff' (obsolete)
biff = no

# no automatic address completion with .a-company.com
append_dot_mydomain = no

# hostname
myhostname = efd.a-company.com

# domain for local e-mails without explicit domain specification
myorigin = a-company.com

# alternatively for Debian/Ubuntu:
# myorigin = /etc/mailname
# /etc/mailname contains a-company.com in the sample configuration

# location of the alias file
alias_maps = hash:/etc/aliases
alias_database = hash:/etc/aliases

# allow sending new emails only from the local computer
mydestination = a-company.com, efd.a-company.com, localhost
mynetworks = 127.0.0.0/8 [::ffff:127.0.0.0]/104 [::1]/128

# no email forwarding to other hosts (no relaying)
relayhost =

# email reception via all network interfaces
inet_interfaces = all

# use IPv4 and IPv6
inet_protocols = all

# no limitation of email and mailbox size
mailbox_size_limit = 0
```

In addition to the keywords included in the preceding listing, there are countless others documented in `man 5 postconf`. All options have default settings. The `postconf -d` command shows you what these look like.

In the following list, I will briefly introduce some particularly important settings. A description of various other options can be found later in this chapter:

- `myhostname` contains the hostname of the server. `myhostname` is considered the default for many other options.
- `myorigin` specifies the domain to which locally sent emails are to be assigned; in the example of this chapter, that's `a-company.com`. In the configuration file supplied with Ubuntu or Debian, `myorigin` is read from the `/etc/mailname` file. Make sure that the correct name is included there, or change the setting in the `main.cf` file!
- `mydestination` lists domains for which received emails are to be delivered (i.e., stored) locally in a mailbox. The default configuration on RHEL uses various variables for `mydestination`. However, `$myorigin` is missing there, which causes the *Relay access denied* error message to appear. If you want to stick to the default syntax, you should change the line as follows:

  ```
  mydestination = $myorigin, $myhostname, localhost.$mydomain,
  localhost
  ```

  Caution: Even if Postfix is responsible for multiple domains, `mydestination` must only contain entries for the primary domain. You can specify virtual domains via the `virtual_alias_domains` option.
- `mynetworks` specifies the addresses from which Postfix accepts mail via SMTP without authentication. The addresses or address ranges specified here thus designate the computers that are "trusted" by Postfix.

  In the configuration presented here (i.e., for a standalone email server on a root server), `mynetworks` must contain only the IPv4 or IPv6 address of `localhost`. If you configure `mynetworks` incorrectly (not restrictively enough), external users can use your mail server to send emails without authentication. Beware, spammers love such computers!

  If `mynetworks` is not explicitly set (e.g., on RHEL), Postfix automatically uses only localhost addresses, so it is a reasonable default setting. You can verify this by using `postconf -d mynetworks`.
- `relayhost` specifies to which MTA emails should be forwarded that are *not* intended for local delivery. In the configuration presented here, `relayhost` must remain empty. Only if Postfix runs on a computer in a local network and is supposed to forward emails to be sent to an external MTA on the internet is `relayhost` is the decisive parameter.
- `inet_interfaces` controls at which network interfaces Postfix waits for incoming emails. The default `all` setting means that all interfaces are active. `loopback_only` or the specification of the `127.0.0.1` and `::1`

addresses means that only server-internal network traffic is processed. Changes to this option require a restart of Postfix (`systemctl restart postfix`)!

26.2.5 Changes to the Configuration

Postfix consists of a number of individual programs, many of which are restarted every time they are needed and then terminated again shortly after. These programs load the relevant configuration files anew each time.

However, there are also Postfix components that do not automatically detect configuration changes. Postfix beginners, who often do not know which changes Postfix notices on its own, should always run `systemctl reload postfix` after the configuration has been modified. This is especially true for changes to `master.cf` and `main.cf`.

After resetting `inet_interfaces`, even `systemctl restart postfix` is required. However, Postfix professionals will avoid `reload` and `restart` if possible because of the associated loss of speed, especially on large active systems.

postconf

Instead of changing `main.cf` in an editor and then running a `reload` command, you can also implement the change using `postconf -e option=value`. The `postconf` command then also immediately notifies Postfix about the change. The command changes existing settings in `main.cf` at their previous location, while new parameters are added at the end of the file. Make sure that you do not insert a space before and after the `=` sign:

```
root# postconf disable_vrfy_command=yes
```

Testing the settings

If you aren't sure which default settings apply in Postfix and which settings you have made that differ from the basic settings, the following three commands will help you:

```
root# postconf -d   (default settings)
root# postconf      (all currently valid settings)
root# postconf -n   (only deviations from the basic settings)
```

If you're only interested in a specific parameter, you need to specify it explicitly:

```
root# postconf -d local_recipient_maps
local_recipient_maps = proxy:unix:passwd.byname $alias_maps
```

The additional `-f` option (*fold*) improves the readability of the output by inserting line breaks.

soft_bounce

Once a mail server has been put into operation, changes during operation are obviously a tricky matter. If you do something wrong, incoming emails may be rejected. These are then lost for good. That's why it's better to add `soft_bounce=yes` to `main.cf` before starting the modifications (and of course to keep a sharp eye on the logging files). Due to `soft_bounce=yes`, Postfix converts all `5xx` errors (*permanent failures*) to `4xx` errors (*persistent transient failures*). The sender will therefore make a new delivery attempt after some time and the email is not lost.

Don't forget to set `soft_bounce=no` again after finishing and testing the changed configuration!

IPv6

Postfix uses IPv4 and IPv6 by default. The two relevant options are set accordingly by default:

```
inet_protocols = all
smtp_address_preference = any
```

Only if you want to explicitly disable IPv6 should you use `inet_protocols = ipv4`.

26.2.6 Opening Port 587

By default, Postfix accepts messages only on port 25 for further processing. This port is intended for the communication between mail servers. While most mail clients can be configured to use port 25, port 587 is actually intended for sending such emails.

Postfix already contains the appropriate statements in the `master.cf` file (not in `main.cf` as usual); they only need to be commented out. In `master.cf`, you must look for the line that starts with `submission inet`, and then remove the `#` comment character from that line and subsequent lines that start with `-o` and contain miscellaneous parameters. Only leave the three lines where `$mua variables` are referenced unchanged (unless you have made appropriate `mua_xxx` settings in `main.cf`):

```
# enable port 587 in /etc/postfix/master.cf
#
 service type  private unpriv  chroot  wakeup  maxproc command + args
#               (yes)   (yes)   (yes)   (never) (100)
...
submission inet n       -       n       -       -       smtpd
  -o syslog_name=postfix/submission
  -o smtpd_tls_security_level=encrypt
  -o smtpd_sasl_auth_enable=yes
  -o smtpd_reject_unlisted_recipient=no
```

```
# -o smtpd_client_restrictions=$mua_client_restrictions
# -o smtpd_helo_restrictions=$mua_helo_restrictions
# -o smtpd_sender_restrictions=$mua_sender_restrictions
  -o smtpd_recipient_restrictions=permit_sasl_authenticated,reject
  -o smtpd_relay_restrictions=permit_sasl_authenticated,reject
  -o milter_macro_daemon_name=ORIGINATING
```

In short, these settings cause Postfix to accept messages at port 587, but only if the client can authenticate itself and the communication is encrypted. For the changed configuration to take effect, you must run `systemctl reload postfix`.

26.2.7 Logging and Administration

Logging

Postfix logs its activities via syslog. As is so often the case, the location of the numerous messages depends on the distribution:

- Some distributions contain several logging files in `/var/log`, which, for example, are named `mail.log`, `mail.info`, `mail.warn`, and `mail.err`. This makes it easier to differentiate between more or less important messages.
- It is more common that simply all mail-relevant messages end up in `/var/log/mail.log` or `/var/log/maillog`. You can use `grep` to filter out the messages that concern a specific program (`grep postfix mail.log`), that describe an error (`grep -i error mail.log`), and so on.

  On RHEL, the `/var/log/maillog` file exists only if you install the `rsyslog` package and then run `systemctl enable --now rsyslogd`.
- An increasing number of distributions use the systemd journal for logging tasks. `journalctl -u postfix` then returns all postfix messages, `journalctl -u dovecot` returns those from Dovecot, and so on. However, usually only a few warnings and error messages are logged in the journal. `mail.log` or `maillog`, on the other hand, contain status messages for each mail received or sent.

mailq

If you want to know which emails are waiting to be sent, you should run `postqueue -p` or `mailq`. The two commands provide a list of all emails that—for whatever reason—could not be sent so far.

You can use `postqueue -f` to trigger an immediate retry. (Usually Postfix itself decides when to make the next attempt).

26.3 Postfix Encryption (TLS/STARTTLS)

Postfix supports the TLS protocol and the STARTTLS method for establishing an encrypted connection between itself and the mail client that wants to send an email. However, this assumes that `main.cf` is configured accordingly and that you have a suitable certificate for your machine. Basic knowledge about "correct" and self-generated certificates can be found in Chapter 24, Section 24.2. For more information about Postfix TLS support, visit *https://www.postfix.org/TLS_README.html*.

Note that changes to the TLS configuration will only take effect after a restart of Postfix!

smtpd = Incoming, smtp = Outgoing

When configuring TLS in `main.cf`, there are two groups of parameters you must not confuse. In one of the groups, the parameter names start with `smtpd_`; these parameters control the behavior of Postfix upon receiving emails. This especially concerns the communication between the mail clients of the employees of your company/organization and the mail server.

In the other group, the parameter names start with `smtp_`; these parameters concern the sending of emails—that is, the communication of the local MTA with other MTAs.

Custom certificates are only required for incoming emails (`smtpd_*`)! The mail client or an external MTA asks Postfix for the public part of the key and encrypts the message. Postfix has the private key and can use it to decrypt the incoming message.

When sending emails, it's exactly the other way around: that's when Postfix asks the target MTA if it can provide a public key to encrypt the message. Postfix thus relies on the public part of an external key.

26.3.1 Sample Configuration and Keywords

The following lines reflect a sample configuration:

```
# in/etc/postfix/main.cf
...
# receive mails encrypted if possible;
# certificate, key and, if required,
# certificate chain for the certification authority
smtpd_tls_security_level = may
smtpd_tls_cert_file      = /etc/ssl/my-full-chain.pem
smtpd_tls_key_file       = /etc/ssl/my-key.key
```

```
# smtpd_tls_CAfile       = /etc/ssl/my-ca-bundle.pem

# send mails encrypted if possible
smtp_tls_security_level  = may

# certificate authorities for Debian/Ubuntu
smtp_tls_CApath          = /etc/ssl/certs

# certificate authorities for RHEL
smtp_tls_CApath          = /etc/pki/tls/certs
smtp_tls_CAfile          = /etc/pki/tls/certs/ca-bundle.crt

# optional: higher efficiency through cache
smtpd_tls_session_cache_database = btree:${data_directory}/smtpd_scache
smtp_tls_session_cache_database  = btree:${data_directory}/smtp_scache

# optional: Logging for troubleshooting
smtpd_tls_loglevel = 1
smtp_tls_loglevel  = 1
```

The keywords used here require some further explanation:

- `smtpd_tls_security_level = may` causes Postfix to offer STARTTLS and accept incoming mail encrypted with TLS if possible. However, if the mail client is configured incorrectly or for old mail clients, authentication in plain text is still possible. If you want to force the encryption, you need to specify the `encrypt` string instead of `may`. This guarantees an encryption, but makes it impossible to receive email from poorly configured clients or MTAs.
- `smtpd_tls_cert_file` and `smtpd_tls_key_file` specify which certificate Postfix should use with which key. With certificates from official certification authorities, you usually receive another file that closes the certificate chain. You need to specify this using `smtpd_tls_CAfile`. You can do without this in the case of self-created certificates or certificates from Let's Encrypt.
- `smtp_tls_security_level = may` signifies that Postfix sends emails encrypted if possible, provided that the recipient (i.e., the mail server on the other side) is capable of receiving encrypted mails. If necessary, the message can also be transmitted in plain text. A more secure solution would be `smtp_tls_security_level = encrypt`, but then your mail server can no longer send messages to mail servers that do not support encryption.

- `smtp_tls_CAfile` references a collection of CAs (certificate authorities) Postfix should trust. The directory location for these files varies depending on the distribution.

  On Debian and Ubuntu, Postfix runs in a `chroot` environment and has no access to `/etc/ssl/certs` at all. For this reason, the certificates are automatically copied to the `chroot` directory `/var/spool/postfix/etc/ssl/certs` when Postfix is started.

  Things are much easier on RHEL: there, SELinux monitors the Postfix programs, so a `chroot` environment is not required.

  If `smtp_tls_CAfile` is not set correctly, then the *Untrusted TLS connection* message is logged when sending mails. This means that the message was transmitted encrypted, but Postfix could not verify the certificate of the other mail server.

  With the correct setting of `smtp_tls_CAfile` and assuming that the recipient server's certificate was issued by a CA that is included in the CA collection, the logging entry changes to *Trusted TLS connection*. The following page of the Postfix documentation provides help for interpreting *Untrusted*, *Trusted*, and and the like (see *https://www.postfix.org/FORWARD_SECRECY_README.html#status*).
- Using `smtp[d]_tls_session_cache_database`, Postfix manages a cache that temporarily stores TLS session information. The database is shared by multiple Postfix instances. It significantly speeds up the TLS communication. The cache files are stored in the Postfix data directory, `/var/lib/postfix`.

Default configuration on Debian and Ubuntu

On Debian and Ubuntu, Postfix uses a so-called snake-oil certificate. This self-signed certificate is generated during the installation of Apache or Postfix (depending on which package was installed first) for the host name that was valid at that time and is valid for 10 years (see Chapter 24, Section 24.2.4). The `smtpd_tls_cert_file` and `-key_file` parameters in `main.cf` specify the locations of the certificate and key files:

```
# file /etc/postfix/main.cf
...
smtpd_tls_cert_file = /etc/ssl/certs/ssl-cert-snakeoil.pem
smtpd_tls_key_file  = /etc/ssl/private/ssl-cert-snakeoil.key
```

During the configuration, the mail client will of course report that the certificate is not trusted. In most mail clients, you can still accept the certificate; this will securely encrypt the mails transferred from the mail client to Postfix, and you will have achieved your goal.

While the use of self-signed certificates is an absolute no-go for public web servers (except for test operations), the situation is completely different for mail servers. The highest priority is currently that the data flow gets encrypted at all. That's why most mail servers accept encryption when receiving mail even if the sending server uses a self-signed certificate, if it has already expired, and so on.

Current Postfix installations on Debian and Ubuntu receive email encrypted and send email encrypted. For older installations, however, only the former applies. For this reason, you need to make sure that `main.cf` contains the following two statements:

```
smtp_tls_CApath=/etc/ssl/certs
smtp_tls_security_level=may
```

Default configuration on RHEL

The default configuration on RHEL 8 is almost exactly the same as the listing printed earlier, with self-signed certificates in use as in Debian/Ubuntu.

26.3.2 Setting Up Custom Certificates

Determine valid domain for certificate

Unless your certificate is a wildcard certificate that applies to all `*.a-company.com` subdomains, the question arises: For which domain name should the certificate used by Postfix actually apply? For `a-company.com` or for `efd.a-company.com` (i.e., for the host name according to the MX entry) or maybe for `smtp.a-company.com`? According to *https://serverfault.com/questions/389413*, only the client (e.g., Thunderbird) is responsible for verification. The client, in turn, compares the domain name of the certificate with the host name you specify during the client configuration of the mail account. The MX entry is not relevant for that.

By default, Thunderbird adds `smtp` as a prefix to the host name for the SMTP configuration. This results, for example, in `smtp.a-company.com`. Then the certificate should also have this name. If you continue this consistently, you will need a separate certificate for each mail protocol (`pop.a-company.com`, `imap.a-company.com`, etc.).

Snake-oil certificates (Debian/Ubuntu)

On Debian and Ubuntu, a snake-oil certificate including a key is set up by default during the Postfix installation—that is, a self-signed certificate that's valid for ten years. The `smtpd_tls_cert_file` and `-key_file` parameters in `main.cf` specify the locations of the certificate and key files:

```
# file /etc/postfix/main.cf
...
smtpd_tls_cert_file = /etc/ssl/certs/ssl-cert-snakeoil.pem
smtpd_tls_key_file  = /etc/ssl/private/ssl-cert-snakeoil.key
```

If you need a new certificate after changing the host name, you want to call the following command:

```
root# make-ssl-cert generate-default-snakeoil --force-overwrite
```

However, the snake-oil certificate has two disadvantages: First, it's self-signed, and second, it applies only to the host name, not to `mail.hostname`.

Creating a custom wildcard certificate

On RHEL, you can quickly generate a self-signed wildcard certificate instead of the snake-oil certificate by using the `openssl` command. It's crucial that when you run `openssl`, you prefix your host name with "`*.`" when entering the *common name*—for example, `*.a-company.com`. Don't forget `chmod 600` for the key file:

```
root# mkdir /etc/ssl/private    (RHEL only)

root# openssl req -new -x509 -days 3650 -nodes \
        -out    /etc/ssl/certs/postfix.pem \
        -keyout /etc/ssl/private/postfix.key
...

Common Name (eg server FQDN or YOUR name) []: *.a-company.com
...
root# chmod 600 /etc/ssl/private/postfix.key
```

Then you must modify `main.cf` as follows:

```
# file /etc/postfix/main.cf, use custom certificate
...
smtpd_tls_cert_file = /etc/ssl/certs/postfix.pem
smtpd_tls_key_file  = /etc/ssl/private/postfix.key
```

Using Let's Encrypt certificates

Although the Let's Encrypt project was founded to advance the encryption of websites (see Chapter 24, Section 24.3), there is nothing to prevent the certificates from being used for the mail server as well. The configuration is easiest if you have a wildcard certificate. However, as issuing and updating such certificates works well only in conjunction with some large DNS servers, you will probably use ordinary certificates.

Make sure that the certificate includes the full host name (here, `efd.xxx`) as well as all subdomains common for mail servers, like `mail.xxx`, `smtp.xxx`, `imap..xxx`, and `pop..xxx`. If you have not yet installed the `acme.sh` command, please read the aforementioned Chapter 24, Section 24.3 for instructions:

```
root# acme.sh --issue --server letsencrypt  \
        -d mail.a-company.com -d smtp.a-company.com \
        -d imap.a-company.com -d efd.a-company.com \
        -w /var/www/html
```

This command assumes that the mail server is also running a web server and that its document root directory is `/var/www/html`. If there is no active web server, then replace `-w /var/www/html` with the `--standalone` option. The `acme.sh` script then temporarily acts as a web server itself. This is only possible if port 80 is not blocked by any other program.

Now the certificates must be copied to a directory that can be accessed by Postfix. Because the certificates are also to be used later by the Dovecot program, it's advisable to install this program now and take it into account with the `--reloadcmd` option. The certificates will be renewed every 60 days in the future. For Postfix and Dovecot to then each use the current certificates, these two programs must be restarted:

```
root# mkdir /etc/acme-letsencrypt     (if the directory does not exist yet)
root# apt install dovecot-imapd       (Debian/Ubuntu)
root# dnf install dovecot             (RHEL)
```

```
root# acme.sh --install-cert -d mail.a-company.com \
  --cert-file  /etc/acme-letsencrypt/mail.a-company.com.cert \
  --key-file   /etc/acme-letsencrypt/mail.a-company.com.key \
  --fullchain-file /etc/acme-letsencrypt/mail.a-company.com.fullchain \
  --reloadcmd 'systemctl restart postfix; systemctl restart dovecot'
```

Now you must modify `main.cf` as shown in the following sample listing:

```
smtpd_tls_cert_file = \
  /etc/acme-letsencrypt/mail.a-company.com.fullchain
smtpd_tls_key_file  = /etc/acme-letsencrypt/mail.a-company.com.key
```

Make sure that you set the `*.fullchain` file as `smtpd_tls_cert_file`, not as `*.cert`.

Additional SSL options

Like Apache, Postfix also supports various encryption algorithms. If required, you can use additional options to control which methods are accepted and which are not; this way, you can increase the security of the mail server operation. The website at *https://ssl-config.mozilla.org/#server=postfix* provides excellent assistance in setting the options correctly.

Optimizing the options, however, resembles a tightrope walk: the more recent protocol versions your Postfix configuration requires, the greater the risk of incompatibilities with other servers or mail clients.

Changes to the TLS configuration do not take effect until Postfix has been restarted:

Restart required

```
root# systemctl restart postfix
```

26.4 Postfix Accounts

The following generally applies to Postfix: every regular user in `/etc/passwd` is also a mail user. No additional configuration is required for these users. Each of these Linux users can send and receive emails.

External mail clients such as Thunderbird must authenticate with Postfix or Dovecot. The regular Linux passwords are also used for this authentication. (This is a big difference compared to Samba; this file server maintains separate passwords for each Samba user.)

Mail Password as a Security Risk

As convenient as it is to have Linux and mail users running in parallel by default, it can also be risky.

Probably nobody would be so unwise as to use `root` as a regular email account (i.e., `root@a-company.com`) as this would require you to enter the `root` password when configuring the mail client. Rather, the risk of the following scenario is more likely to be overlooked: As an administrator of the `a-company.com` server, you have an account there with `sudo` rights—for example, `hoover`. For admin tasks, you log in with `ssh hoover@a-company.com`, then run `sudo -s` and do whatever needs to be done. So far, so good.

It now makes sense to also use the `hoover@a-company.com` email account. As explained earlier, no further configuration work is required for this. The problem is that you now enter your `hoover` password once when configuring Thunderbird on your Linux notebook, again when configuring your mail program on the smartphone, again for the tablet, and so on. If one of these devices is lost, it may be possible to find the stored password for `hoover`. And not only is this password sufficient to read your emails or send them in your name, but the dishonest finder can now also log in to `company.abc.com` using `ssh` and has unlimited rights there thanks to `sudo`. In terms of security, this is the worst thing that can happen.

To solve this problem, simply never use mail accounts of users with `sudo` privileges! Use separate accounts for email and for admin work, even if it's inconvenient.

Set up a Linux account for mail use

To set up a Linux account on the mail server for mail use only, you should set `/bin/false` as the shell. This makes it impossible to log in and work interactively.

On the other hand, the password is important, even though the user cannot log in at all. Here's why: the password applies to the POP and SMTP authentication described in Section 26.5.

On Debian and Ubuntu, the easiest way to set up a new account is to use `adduser`. `--gecos ,,,` suppresses the questions for the full name and other superfluous contact information:

```
root# adduser  --shell /bin/false --gecos ,,, hoover   (Debian/Ubuntu)
Enter a new UNIX password:        ********
Enter the new UNIX password again:  ********
```

On RHEL, you need to call `useradd` and `passwd`:

```
root# useradd -s /bin/false hoover                     (RHEL)
root# passwd hoover
Changing password for user hoover.

New password:        ********
Retype new password: ********
```

> **Tip**
>
> Virtual domains, which I will introduce ahead, allow you to avoid setting up a separate Linux account for each user. Postfix then supports virtual mailboxes, which are quite real but do not correspond to a valid user name. Such a configuration is useful if you do not need three or four mail accounts, but hundreds or thousands.

Mail accounts versus Linux user accounts

The parallel connection of Linux and mail accounts outlined in the introduction corresponds to the default configuration, but it will rarely remain that way. In the following sections, I will therefore explain some possibilities for a configuration that deviates from this:

- Mail aliases make it possible to redirect emails to existing accounts.
- By configuring an explicit list of recipients, you can control the list of email accounts and, for example, prevent system accounts such as `adm` or `lp` from sending or receiving emails.
- If you use virtual domains, you can process emails from different host names using one Postfix server (not only for `a-company.com`, but also for `company-xy.info`).

- Virtual mailboxes allow you to completely separate mail accounts and their passwords from Linux accounts. The data is often stored in a database.

First, however, the following section deals with the question of where emails are actually stored.

26.4.1 mbox or maildir Format

By default, Postfix stores incoming emails in mbox format in the `/var/mail/name` file. If emails are to be stored in a local file in the user directory instead (still in mbox format), you must specify the desired file name relative to the home directory by using `home_mailbox`:

```
# in /etc/postfix/main.cf
...
# save emails in mbox format in file /home/name/Mailbox
home_mailbox = Mailbox
```

If you plan to retrieve the mails mainly via IMAP, you should prefer using the maildir format. The correct setting of `home_mailbox` now looks as follows:

```
# in /etc/postfix/main.cf
...
# save emails in maildir format in the /home/name/Maildir directory
home_mailbox = Maildir/
```

The directory name must end with / so that Postfix uses the maildir format. If `main.cf` contains a `mailbox_command` statement, you must comment it out. New incoming emails are now stored in their own files in the `/home/<name>/Maildir` directory.

maildir for root on RHEL

Basically, the conversion to the maildir format should work without any problem. However, the `root` user on RHEL systems does cause problems. In this case, the mail delivery fails with the following error: *create maildir file: permission denied*. Fortunately, `/var/log/maillog` also immediately points out the cause. For security reasons, Postfix refuses to set up the required maildir directories for `root`. You must take care of this yourself in advance:

```
root# mkdir -p /root/Maildir/{cur,new,tmp}
```

maildir with Dovecot and Mutt

Don't forget to tell your local mail client or Dovecot the location and format of your mailbox as well (Section 26.5)! If you use `mutt`, you must configure the program accordingly so that it reads the emails from the `maildir` directory (see Chapter 11, Section 11.5).

26.4.2 Mail Aliases

A *mail alias* is an additional mail name for receiving email. However, the email is actually forwarded to an existing account. Aliases are defined in the `/etc/aliases` file. This file usually looks similar to the following pattern:

```
# file /etc/aliases
postmaster:      root
webmaster:       hoover
Herbert.Hoover:  hoover
...
```

The first column specifies the alias name without a domain. The name applies to the domain defined in `myhostname`, such as `postmaster@a-company.com`. The second column contains the local recipient. In the preceding example, emails addressed to `postmaster` are forwarded to `root`, while emails to `webmaster` and to `Herbert.Hoover` are forwarded to `hoover`. It's permissible to specify multiple recipients separated by commas for each alias in `/etc/aliases`.

As a recipient, you can specify an external email address instead of a local email account name, a file to which the email will be attached, or a program in the `|command` notation to which the email will be redirected. Redirecting to external email addresses works without any problem, but in practice often fails due to the spam protection of the target MTA. So, for example, if you redirect emails to `webmaster` at `name@gmx.com`, the spam filter of GMX will recognize that the email was not transmitted directly to `gmx.com`, but indirectly via `a-company.com`. This is enough for a suspicious spam filter to classify the email as spam.

For changed aliases to take effect, you must run the `newaliases` command. This command synchronizes the `/etc/aliases` text file with the `/etc/aliases.db` BDB file.

alias_database and alias_maps

In `/etc/postfix/main.cf`, you usually encounter two `alias` options, which sometimes causes confusion:

```
# in /etc/postfix/main.cf
...
alias_maps = hash:/etc/aliases
alias_database = hash:/etc/aliases
```

`alias_database` specifies which database file is to be updated by the `newaliases` command. The file specified here (whose name is supplemented by `.db`) contains the aliases specified in `/etc/aliases` in a binary format.

alias_maps specifies which alias databases Postfix is supposed to take into account. Typically you specify the same file here as for alias_database. However, it is permissible to specify additional sources for alias definitions. The crucial difference between the two parameters is that alias_database applies to newaliases, while alias_maps applies to Postfix!

Cleaning up Aliases on RHEL

In the default configuration, /etc/aliases on RHEL contains an almost endless list of aliases that are rarely useful in real life. Create a backup of the file, and then delete all the lines you don't currently need!

.forward

Even as a local user who does not have access to /etc/aliases, you can redirect your mail to another address: to do this, you should create the .forward file and save the new destination address in it. Done!

As I mentioned previously, this simple method of email redirection usually only works as planned for local addresses. With external destination addresses, on the other hand, it can happen that the redirected emails are caught in the spam protection of the destination MTA.

26.4.3 Explicit Recipients List

By default, any Linux account can receive email. However, it's rarely desirable for system accounts such as daemon, sys, or man to receive emails. To fix this issue, you need to tweak the **local_recipient_maps** parameter. The default setting is as follows:

```
local_recipient_maps = proxy:unix:passwd.byname $alias_maps
```

This means that all users listed in the Unix /etc/passwd file and all users named in the alias databases can receive email. If you want only fisher, hoover, smith, and root (for system notifications) to receive emails, you should proceed as follows: First, create a text file that contains the account names line by line. The file must contain an arbitrary value in a second column, because Postfix and the postmap command generally expect *key/value pairs* even for lists where actually only the existence of a key is relevant, but the corresponding value is not considered at all:

```
# file /etc/postfix/local-recips
fisher   x
hoover   x
smith    x
root     x
```

From this file, you then create a Postfix-readable database file named `local-recips.db` by using `postmap`:

```
root# cd /etc/postfix
root# postmap local-recips
```

The `postmap -s` command enables you to check if the file was created correctly:

```
root# postmap -s hash:local-recips
hoover  x
root    x
smith   x
fisher  x
```

Then, add a line to `/etc/postfix/main.cf` to set `local_recipient_maps`:

```
# in /etc/postfix/main.cf
local_recipient_maps = hash:/etc/postfix/local-recips $alias_maps
```

From now on, Postfix will only accept emails to the recipients named in `local-recips`. However, there is still one limitation: emails are only delivered (i.e., stored locally) if the user has an account on the computer. If that is not the case, you have to create a new account using `adduser`.

Updating Mapping Files Safely and Securely

After each change to `local-recips`, you must run `postmap` to update the `local-recips.db` database file relevant to Postfix. However, this update is a critical operation: `postmap` deletes `local-recips.db` and then rewrites the file. If Postfix accesses `local-recips.db` just during this time, it will receive incorrect or incomplete data.

A safe approach therefore looks as follows: you assign a different name to the underlying text file (e.g., `local-recips1`), apply `postmap` to this file and then run `mv local-recips1.db local-recips.db`. Thus, `local-recips.db` contains consistent data at any time—either in the old or in the new version. You can automate this process by using a script or `make` file, as described at *https://www.postfix.org/DATABASE_README.html#safe_db*.

This note applies to *all* mapping files that appear in this chapter.

26.4.4 Email Addresses Differing from the Linux Account

In many companies, email addresses of the format `firstname.familyname@company.com` are common. Linux, however, doesn't provide for such long

user names—not to mention those with a dot. However, this does not prevent you from still using long email names:

- Create a Linux account with a Linux-typical, short user name, such as `hoover`.
- Define an alias rule that forwards emails sent to `Herbert.Hoover` to `hoover`.
- When configuring the email client, use the long format as the sender address—for example, `Herbert.Hoover@a-company.com`. Note, however, that you must specify the Linux account name for POP, IMAP, and SMTP authentication!
- To ensure that emails sent locally (e.g., by Mutt) also contain the correct sender address in the long version, you need to set up a new table in the `/etc/postfix/canonical` text file and convert it to a file readable by Postfix using `postmap`. The table specifies how Postfix is supposed to modify email addresses:

```
# /etc/postfix/canonical
hoover Herbert.Hoover@a-company.com
...
```

- Then, in `main.cf`, set the `canonical_maps` parameter, and then run `systemctl reload postfix`:

```
# in /etc/postfix/main.cf
...
canonical_maps = hash:/etc/postfix/canonical
```

In addition to the Canonical and alias tables, Postfix provides various other ways to manipulate email addresses at different stages of email traffic: when receiving, before sending, and so on. The page at *https://www.postfix.org/ADDRESS_REWRITING_README.html* gives a good overview.

26.4.5 Virtual Domains with Shared Email Users

In the simplest case, Postfix is only responsible for emails sent to the host name of the computer, such as `xxx@a-company.com`. However, it's often useful for *one* MTA to be responsible for multiple email domains, such as `xxx@another-company.com`, for example. All domains that do not match the host name of the computer are called *virtual* in Postfix terminology (they're often called *hosted domains* as well).

Postfix provides several ways to implement virtual domains. The easiest way is to simply set several domains in the `mydestination` parameter, like this:

```
# in /etc/postfix/main.cf
...
mydestination = a-company.com, localhost, another-company.com
```

Of course, you also need to adjust the DNS configuration of `another-company.com` accordingly. The MX host name must therefore be assigned the IP address of your server (see Section 26.1.5).

This ensures that mails to `another-company.com` are treated in the same way as mails to `a-company.com`. So it doesn't matter whether an email is addressed to `hoover@a-company.com` or to `hoover@another-company.com`: in any case, Postfix delivers the email to the local user `hoover`. For some cases, this is sufficient—especially if a company or organization has multiple web presences, but in reality the same people are always responsible for them.

26.4.6 Virtual Domains with Separate Email Users

If you want to differentiate between users with the same name depending on the domain, you must specify the domains involved using the `virtual_alias_domains` keyword rather than `mydestination`. In addition, you then need a table that establishes the mapping between the virtual domain email addresses and the real Linux accounts. Furthermore, a Linux account is required for each email address. To stick with the example of the hoover addresses: The account for `hoover@a-company.com` is still `hoover`. For `hoover@another-company.com`, you have to create a new account—for example, `hooverNEF`. The structure of the table for the virtual email users looks as follows:

```
# text file /etc/postfix/virtual
hoover@another-company.com      hooverNEF
miller@another-company.com    millerNEF
...
```

Like the alias table, the table can assign the same user to multiple email addresses—for example:

```
# in /etc/postfix/virtual
webmaster@another-company.com  hooverNEF
```

You can use `postmap` to turn this file into a database file that can be read by Postfix:

```
root# postmap /etc/postfix/virtual
```

Now you need to adjust `main.cf`. `virtual_alias_domains` lists all virtual domains (but not the primary domain; you still have to specify that using

mydestination!). virtual_alias_maps specifies the filename of the virtual alias table:

```
# in /etc/postfix/main.cf
mydestination         = a-company.com, localhost
virtual_alias_domains = another-company.com, company-xyz.com, ...
virtual_alias_maps    = hash:/etc/postfix/virtual
```

26.4.7 Virtual Domains with Virtual Mailboxes

Up to this point, it has always been necessary to have a Linux account for each email address. Postfix refuses to store email in a mailbox unless there is a Linux account with the same name. However, if you have a large number of email addresses, it becomes increasingly impractical to create a new account each time. Postfix provides virtual mailboxes to solve this problem. These are ordinary mailbox files; the term *virtual* only refers to the fact that there are no associated Linux accounts.

Of course, the virtual mailboxes also have to belong to someone. For this purpose, you want to create a new group and a new user with unused GIDs and UIDs; in the following example, they are each 5000:

```
root# groupadd -g 5000 vmail
root# useradd -g vmail -u 5000 vmail -d /home/vmail -m
```

Now you modify main.cf as in the following sample listing. virtual_mailbox_domains specifies the virtual domains whose emails are to be stored in virtual mailboxes. virtual_mailbox_base specifies the directory in which the virtual mailboxes are to be created. The virtual_mailbox_maps table establishes the mapping between the email addresses and the mailboxes. virtual_uid_maps and virtual_gid_maps specify the UID and GID of the mailbox files. In theory, it's possible to specify separate UIDs and GIDs for each mailbox, but this is rarely useful:

```
# in /etc/postfix/main.cf
mydestination           = a-company.com, localhost
virtual_mailbox_domains = another-company.com, company-xyz.com, ...
virtual_mailbox_base    = /var/mail
virtual_mailbox_maps    = hash:/etc/postfix/virtual-mboxes
virtual_uid_maps        = static:5000
virtual_gid_maps        = static:5000
```

The virtual-mboxes file specifies the associated mailbox file for each email address relative to the virtual_mailbox_base path. This example uses a separate directory for each domain and then simply the user name within that

directory. Basically, however, you can proceed as you wish at this point. The Postfix `virtual` command is responsible for the actual delivery. It saves the emails in mbox format by default. If you prefer the maildir format, you simply need to add a slash after the filename in `virtual-mboxes`, such as `another-company.com/hoover/`:

```
# file /etc/postfix/virtual-mboxes
hoover@another-company.com    another-company.com/hoover
miller@another-company.com    another-company.com/miller
webmaster@company-xyz.com     company-xyz.com/webmaster
...
```

`postmap` turns `virtual-mboxes` into a database file:

```
root# postmap /etc/postfix/virtual-mboxes
```

The last step is to create the mailbox directory for each domain. The access rights are decisive here:

```
root# mkdir /var/mail/another-company.com
root# chown vmail:mail /var/mail/another-company.com
root# chmod g+w /var/mail/another-company.com
```

`systemctl postfix reload` activates the configuration. Now you should send a test message to `hoover@another-company.com` and then take a look at the `/var/mail/another-company.com/` directory. The `hoover` file should now appear there with the new email.

If you want to manage *all* domains virtually, including the domain of your computer's host name, you must remove the host name from the `mydestination` line and add it to the `virtual_mailbox_domains` line:

```
# in /etc/postfix/main.cf
mydestination           = localhost
virtual_mailbox_domains = a-company.com, another-company.com, ...
```

Managing Virtual Mailboxes Using MySQL Tables

Although virtual mailboxes avoid the need to set up an account for each email address, they make configuring Dovecot to collect emails (POP) and SMTP authentication more complicated as you now need to administrate another table that contains a login name and password for each user.

Virtual mailboxes reduce the administration effort only if you simultaneously store all the data of the email accounts in a database or in an LDAP system and Postfix and Dovecot can access this database equally. Detailed

instructions on how to do this using MySQL can be found on the following pages:

- *https://workaround.org/ispmail/bullseye*
- *https://computingforgeeks.com/how-to-setup-mail-server-on-centos*

26.4.8 Disabling the Address Verification (VRFY)

Among other things, the SMTP protocol provides the `VRFY` command, which can be used to verify the existence of a mail address:

```
user$ telnet localhost 25
Trying 127.0.0.1...
220 a-company.com ESMTP Postfix
VRFY hoover@a-company.com
252 2.0.0 hoover@a-company.com

VRFY bla@a-company.com
550 5.1.1 <hoover@a-company.com>: Recipient address rejected:
User unknown in local recipient table
```

In principle, this is a practical thing. Unfortunately, however, this feature is abused by spammers to identify valid email addresses. For this reason, it's often recommended to disable the verify command. To do this, you need to add a line to `main.cf`:

```
# in /etc/postfix/main.cf
disable_vrfy_command=yes
```

26.5 Dovecot (POP and IMAP Server)

So far we have only installed Postfix. If you run Mutt or a similar email client directly on the server, you can send and receive emails. But if you want to work with external clients, such as Thunderbird on a Linux notebook computer or mail apps on a smartphone, they must be able to access the mail server's mailboxes.

Postfix doesn't provide any functions for this; as a mail server, it only handles the SMTP protocol. The IMAP protocol has been designed to read and manage email on the server, while the POP protocol is hardly used anymore today. It allows you to retrieve emails from the server and transfer them to a local computer. The Dovecot program is responsible for these two protocols.

Installation On RHEL and Fedora, Dovecot and all its features are in the `dovecot` package. On Debian and Ubuntu, on the other hand, Dovecot is distributed across several packages. Depending on which protocol you want to use, you must install `dovecot-imapd` and/or `dovecot-pop3d`.

Firewall The POP3 and IMAP protocols use the ports listed in Table 26.4.

Port	Protocol
Port 110	POP3 with STARTTLS encryption or without encryption
Port 995	POP3S with encryption
Port 143	IMAP with STARTTLS encryption or without encryption
Port 993	IMAPS with encryption

Table 26.4 Firewall Ports for POP and IMAP

Encryption is mandatory for POP3S and IMAPS protocols. In contrast, the POP3 and IMAP protocols were subsequently extended to include the STARTTLS functions, where communication starts unencrypted but continues encrypted where possible. On Fedora, RHEL, and SUSE, all POP and IMAP ports are blocked by default. On SUSE, the easiest way to open the ports is via YaST. On both Fedora and RHEL, on the other hand, you need to proceed as follows, whereby you naturally only release the ports you actually want to use:

```
root# firewall-cmd --get-active-zones   (determine active zone)
public
  interfaces: enp1s0
root# firewall-cmd --permanent --zone=public --add-service=pop3s
root# firewall-cmd --permanent --zone=public --add-service=imaps
root# firewall-cmd --permanent --zone=public \
  --add-port=110/tcp    (POP3)
root# firewall-cmd --permanent --zone=public \
  --add-port=143/tcp    (IMAP)
root# firewall-cmd --reload
```

Configuration In the simplest case, Dovecot works without any configuration work: you wouldn't think it possible that such a thing still exists! In reality, however, a look into the configuration files and various small changes will not be completely avoidable.

There are lots of files for configuring Dovecot, in `/etc/dovecot/dovecot.conf` and `/etc/dovecot/conf.d/*.conf`. The `10-auth.conf` file also contains some `include` statements for `auth-*.ext` files with additions. However,

most `include` statements are commented out by default and therefore not active:

```
user$ cd /etc/dovecot/conf.d
user$ ls
10-auth.conf       15-lda.conf                  auth-deny.conf.ext
10-director.conf   15-mailboxes.conf            auth-dict.conf.ext
10-logging.conf    20-imap.conf                 auth-master.conf.ext
10-mail.conf       90-acl.conf                  auth-passwdfile.conf.ext
10-master.conf     90-plugin.conf               auth-sql.conf.ext
10-ssl.conf        90-quota.conf                auth-static.conf.ext
10-tcpwrapper.conf auth-checkpassword.conf.ext  auth-system.conf.ext
```

The `*.conf` files are also used to document Dovecot. On the one hand, this is practical, but on the other hand, it is also very confusing. By default, the `*.conf` files contain between 1,500 and 1,800 lines of code, depending on the distribution! The actually relevant statements are contained in just a few lines, while the rest are comments.

You can get an overview using the commands ahead. The two nested `grep` commands eliminate all comments and empty lines and display the remaining statements on the screen (the configuration files remain unchanged):

```
user$ cd /etc/dovecot
user$ grep -h -v '^[[:space:]]*\#' dovecot.conf conf.d/*.conf | \
        grep -v '^[[:space:]]*$' | less
```

The following lines present the settings of the most important configuration files on Ubuntu 22.04. The default configuration on RHEL 9 is quite similar, except that `10-ssl.conf` has some small differences. For reasons of space, I have refrained from printing the files that contain only comments but no active code:

```
# file dovecot.conf
!include_try /usr/share/dovecot/protocols.d/*.protocol
dict {
}
!include conf.d/*.conf
!include_try local.conf

# file conf.d/10-auth.conf
auth_mechanisms = plain
!include auth-system.conf.ext
```

```
# file conf.d/10-director.conf
service director {
  unix_listener login/director {
  }
  fifo_listener login/proxy-notify {
  }
  unix_listener director-userdb {
  }
  inet_listener {
  }
}
service imap-login {
}
service pop3-login {
}
protocol lmtp {
}

# file conf.d/10-mail.conf
mail_location = mbox:~/mail:INBOX=/var/mail/%u
namespace inbox {
  inbox = yes
}

mail_privileged_group = mail
protocol !indexer-worker {
}

# file conf.d/10-master.conf
service imap-login {
  inet_listener imap {
  }
  inet_listener imaps {
  }
}
service pop3-login { … }
service submission-login { ... }
service lmtp { ... }
service imap {
}
service pop3 {
}
service auth {
  unix_listener auth-userdb {
```

```
  }
}
service auth-worker {
}
service dict {
  unix_listener dict {
  }
}

# file conf.d/10-ssl.conf
ssl               = yes
ssl_cert          = </etc/dovecot/private/dovecot.pem
ssl_key           = </etc/dovecot/private/dovecot.key
ssl_client_ca_dir = /etc/ssl/certs
ssl_dh            = </usr/share/dovecot/dh.pem

# file conf.d/15-lda.conf
protocol lda {
}

# file conf.d/15-mailbox.conf
namespace inbox {
  mailbox Drafts {
    special_use = \Drafts
  }
  mailbox Junk {
    special_use = \Junk
  }
  ...
}

# file conf.d/20-imap.conf
protocol imap {
}
```

Note that you must close the curly brackets on a separate line. The `protocol imap { }` statement in only one line is syntactically not allowed.

Configuration test

The reason there are so few concrete settings in the Dovecot configuration files is that there are default values for all options. `dovecot -a` returns a list of all currently valid settings, and `dovecot -n` returns a list of all options that differ from the default setting.

IPv6

Dovecot uses IPv4 and IPv6 by default:

```
root# dovecot -a | grep 'listen '
listen = *, ::
```

* means that all IPv4 interfaces are taken into account, and : : applies analogously to IPv6 interfaces. If you want to use IPv4 only, you need to include the following statement in `dovecot.conf`:

```
# file dovecot.conf, disable IPv6
...
listen = *
```

Location of mailbox files

Dovecot copes with both maildir and mbox formats, whether you use the program as a POP or an IMAP server (or both). However, if you use Dovecot mainly as an IMAP server, you should prefer the maildir format for efficiency reasons.

Dovecot tries to discover the mailboxes automatically, which worked fine in my tests. It searches the following directories in this order:

```
/home/username/Maildir                          (maildir format)
/home/username/mail  and  /var/mail/username    (mbox format)
/home/username/Mail  and  /var/mail/username    (mbox format)
```

However, the automatic mailbox search may fail if a user has not yet received any mail and their mailbox is therefore empty—that is, the mailbox file doesn't yet exist. For this reason, it's recommended to explicitly set the mailbox location in `10-mail.conf`. For mailboxes in `/var/mail/name`, the correct setting is as follows:

```
# in /etc/dovecot/conf.d/10-mail.conf
...
# mbox mailboxes in /var/mail
mail_location = mbox:~/Mail:INBOX=/var/mail/%u
```

This tells Dovecot that your server uses the mbox format and that new emails are located in `/var/mail/username`. The additional specification of the `Mail` directory is required for various additional functions of Dovecot—even if there are no emails in this directory! If Dovecot is used as an IMAP server, all other IMAP mailboxes will be stored there. You can of course specify any other user directory, but `mail` or `Mail` are the usual choices. The user directory must be specified before `INBOX`—that is, the mailbox for new emails.

If you have configured Postfix in such a way that the MTA delivers mail in maildir format to the local `Maildir` directory, the correct `mail_location` setting looks as follows:

```
# in /etc/dovecot/conf.d/10-mail.conf
...
# maildir mailboxes in ~/Maildir
mail_location = maildir:~/Maildir
```

If Postfix is configured to store email addresses from various domains in virtual mailboxes, the configuration gets more complicated. First you need to tell Dovecot where the mailboxes are located, and then you need a separate table that contains login names and passwords for POP and SMTP authentication for all email addresses. Configuration tips and a specific example can be found on the following web pages:

- *https://wiki2.dovecot.org/VirtualUsers*
- *https://workaround.org/ispmail/bullseye*

Maximum number of connections

By default, Dovecot allows a maximum of 10 simultaneous connections from one mail client. This seems to be more than sufficient. In real life, however, I have experienced errors occasionally occurring with some email clients (specifically, Mail on macOS), such as *Maximum number of connections from user and IP exceeded*. The solution in such cases is a higher value for the `mail_max_userip_connections` parameter:

```
# file /etc/dovecot/conf.d/20-imap.conf
protocol imap {
  ...
  mail_max_userip_connections = 25
}
```

26.5.1 Operation as POP or IMAP Server

Dovecot works right away as a POP or IMAP server. To download your emails from an external computer with an email client, you want to set up the mail account in the client and specify STARTTLS as the encryption method. The username corresponds to the name of your Linux account on the server. The password is also the same as on the server.

Encryption (TLS/STARTTLS)

Dovecot supports the TLS protocol and the STARTTLS method for establishing an encrypted connection. By default, Dovecot on Ubuntu uses the custom `/etc/ssl/certs/dovecot.pem` certificate, which you must accept in the mail client the first time you connect. On RHEL, there are equivalent certificates in `/etc/pki/dovecot/certs/dovecot.pem`.

If you want to use your own certificate and key, you must adjust the locations of the certificate and key files in `10-ssl.conf`. If you have used `acme.sh` to generate matching Let's Encrypt certificates (see Section 26.3), a matching Dovecot configuration looks as follows:

```
# in /etc/dovecot/conf.d/10-ssl.conf
...
ssl_cert = </etc/acme-letsencrypt/mail.a-company.com.fullchain
ssl_key  = </etc/acme-letsencrypt/mail.a-company.com.key
```

26.5.2 SMTP Authentication for Postfix

Postfix supports the SASL protocol, but it cannot perform the authentication itself. Dovecot can help Postfix in this matter.

What Is SMTP Authentication For?

You may not be fully aware of what SMTP authentication is all about. SMTP servers readily accept email for sending from local programs, such as a web server. This is also true for Postfix: if you run Mutt on the same computer as Postfix, you can use it to compose and send an email, with Postfix taking care of the actual sending.

But if the mail client is running externally—at home on your notebook, on the road on your smartphone—then you need to authenticate with the mail server before you can send an email. This authentication is what we are talking about here.

Dovecot configuration

The required configuration effort is pretty low. First, in the `10-master.conf` configuration file of Dovecot, you must enable the code already provided in the `service auth` section to authenticate via the `/var/spool/postfix/private/auth` socket file:

```
# Additions to /etc/dovecot/conf.d/10-master.conf
...
service auth {
  unix_listener auth-userdb {
    mode  = 0600
    user  = postfix
    group = postfix
  }

  # Postfix smtp-auth
  unix_listener /var/spool/postfix/private/auth {
    mode = 0666
```

```
  }
  # Auth process is run as this user.
  user = $default_internal_user
}
```

Second, in `10-auth.conf`, you need to add the `login` authentication mechanism. This addition is necessary so that authentication works with all common mail clients:

```
# Additions to /etc/dovecot/conf.d/10-auth.conf
...
auth_mechanisms = plain login
```

Postfix configuration

Finally, you need to add some lines at the end of the `/etc/postfix/main.cf` Postfix configuration file. Note that on Debian and Ubuntu, the path for `smtpd_sasl_path` must be relative to the `/var/spool/postfix` directory. This is because, for security reasons, Postfix runs in a `chroot` environment and interprets *all* path specifications in `main.cf` relative to the Postfix queue directory:

```
# addition to /etc/postfix/main.cf
...
smtpd_sasl_auth_enable       = yes
smtpd_sasl_type              = dovecot
smtpd_sasl_path              = private/auth
smtpd_recipient_restrictions = permit_mynetworks,
                               permit_sasl_authenticated,
                               reject_unauth_destination
```

Enabling the configuration

Now, prompt both services to reread their configuration files:

```
root# systemctl restart dovecot
root# systemctl reload  postfix
```

26.6 Client Configuration

"Real" tests with a mail client cannot begin until the Postfix and Dovecot configuration has been completed. The following points are important during configuration (see Figure 26.3):

- Use IMAP (not POP).
- Enable **STARTTLS** for both SMTP (outgoing mail) and IMAP (which is often incorrectly called *incoming mail*).
- If you use a self-signed certificate, you must explicitly accept it—often not during configuration, but the first time you establish a connection.

Some email clients behave quite “intelligently” when it comes to mail configuration. They try common domain names and ports and use the settings that are most likely to fit. This usually works quite well with Thunderbird in particular.

Do not be confused if the configuration refers to plain text passwords (in Figure 26.3, see **Authentication Method**: **Normal Password**). Provided you use the STARTTLS encryption method, plain text passwords are secure because the password transfer takes place within a TLS session, so it is already encrypted at a higher level.

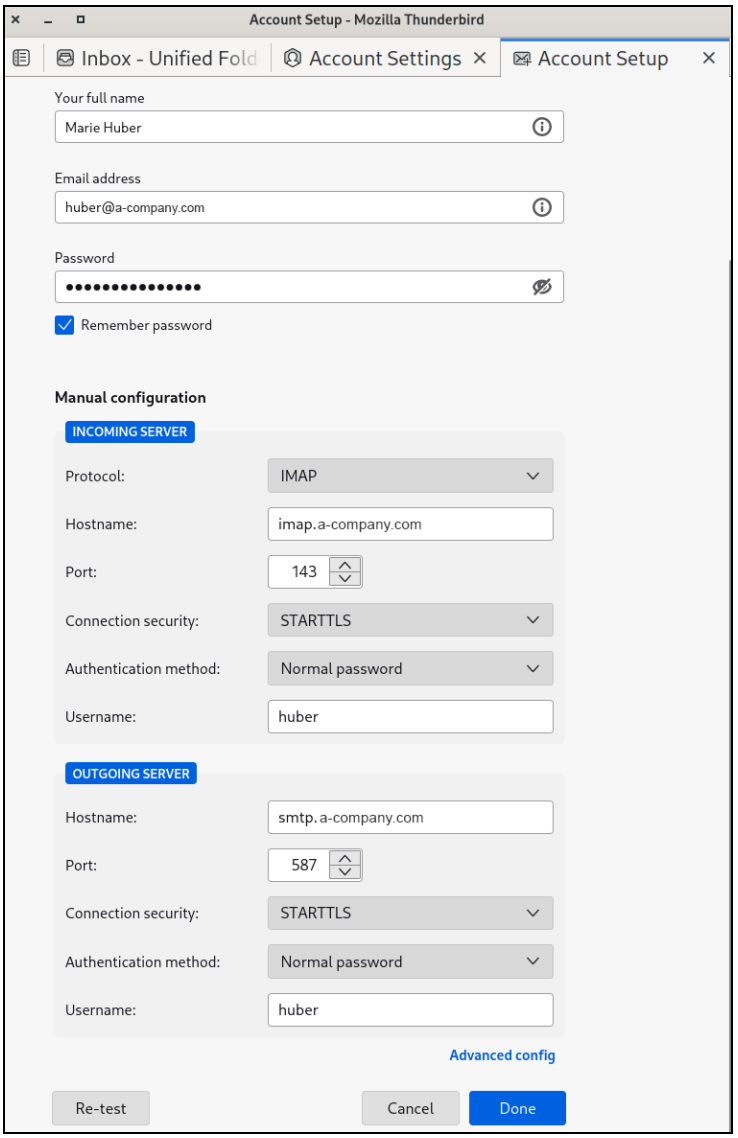


Figure 26.3 Sample Configuration of Mail Account in Thunderbird

26.7 SpamAssassin

Defense against spam is a battle against windmills that can probably never be won. In this section, I will show you how you can at least reduce the flood of spam by using the *SpamAssassin* program. SpamAssassin tries to decide whether an email contains spam or not based on various criteria. Reasons that lead to a spam classification include missing reverse DNS entries, obviously incorrect DNS details, or the origin of emails from known spammers—that is, from computers that have ended up on blacklists due to mass sending of spam.

SpamAssassin also takes the *content* of the email into account. The latter requires a relatively complex analysis, which causes a high CPU load on busy email servers. The results of the various spam detection rules are added together. If the total exceeds a threshold, the email is marked as spam.

Don't get your hopes up too high: On the one hand, SpamAssassin regularly fails to detect obvious spam. But on the other hand, it happens again and again (fortunately much less frequently) that proper emails are classified as spam. In a nutshell, SpamAssassin is better than no spam protection at all, but it is not a miracle cure.

Dealing with emails suspected of being spam

By default, SpamAssassin repackages emails detected as spam. The modified email will then contain a reference to the suspected spam. The original message is included in the attachment so that the user can easily read an email mistakenly classified as spam.

Now the question remains as to how the end user should best deal with emails suspected of being spam. SpamAssassin inserts a line containing `X-Spam-Flag: YES` into each such email.

In addition, each email checked by SpamAssassin gets a line that reads *X-Spam-Level: *********, where the number of asterisks indicates the spam rating. The more asterisks, the higher the spam probability. By default, SpamAssassin considers emails with five or more asterisks as spam.

Because of the `X-Spam` lines in the email header, you can set up a filter rule in most email clients so that emails marked this way are automatically moved to a junk or spam folder.

Later in this section, I will also show you how you can perform the move to a junk folder on the server. Unfortunately, the additional configuration required for this is quite complicated.

Configuration

There are several ways to combine SpamAssassin with Postfix. In the variant presented here, which works on Debian/Ubuntu as well as RHEL, the integration of SpamAssassin is done via the `master.cf` file.

First you need to install the necessary packages:

```
root#  apt install spamassassin spamc     (Debian/Ubuntu)
root#  dnf install spamassassin           (RHEL)
```

The basic configuration of SpamAssassin is carried out by various `*.cf` files in the `/usr/share/spamassassin` directory. Settings that deviate from this are best made in the `/etc/[mail]/spamassassin/local.cf` file:

```
# File /etc/spamassassin/local.cf       (Debian/Ubuntu)
# File /etc/mail/spamassassin/local.cf  (RHEL)
required_score   5.0
rewrite_header   Subject [SPAM]
report_contact   postmaster@a-company.com
report_safe      1
```

Here's a brief explanation of the four settings:

- Probably the most interesting setting is `required_score` (default value 5.0). This indicates the number of points above which an email is classified as spam. If too many correct mails are detected as spam, you should increase the value.

  Previously, the parameter was called `required_hits`. This name is still accepted, but is now considered obsolete.
- Using `rewrite_header Subject xxx`, the text `xxx` is inserted in the subject line of every mail recognized as spam. This makes spam detection easier for email users who are overwhelmed with defining filter rules in their email client.
- `report_contact` specifies a contact address for the spam system. The address is included in the preamble of a spam-suspicious mail.
- `report_safe` specifies what should happen to mails that have been recognized as spam:
  - 0 (default on RHEL): Only an invisible header of the `spam` type is added to the mail.
  - 1 (default on Debian/Ubuntu): The original mail is repackaged and attached to a warning mail.
  - 2: Like 1, but the original mail is only added as text (`text/plain` instead of `message/rfc822`). However, this makes it difficult to read the email.

Before enabling SpamAssassin as a daemon, you need to make two changes in `/etc/default/spamassassin` on Debian/Ubuntu:

```
# change in /etc/default/spamassassin (Debian/Ubuntu only)
# perform regular updates of SpamAssassin rules
CRON=1
```

After this preparatory work, you can start SpamAssassin:

```
root# systemctl enable --now spamassassin
```

Integration with Postfix

To integrate SpamAssassin with Postfix, you need to create a new user on RHEL, which I have here named `spamd`:

```
root# useradd spamd -s /bin/false -d /var/log/spamassassin    (RHEL)
```

On Debian and Ubuntu, this step is omitted. There the `debian-spamd` account was already set up during the installation of `spamassassin`.

To enable Postfix to recognize spam, you must change the existing line in `/etc/postfix/master.cf` that begins with `smtp` and add another statement. In the following listing, this statement is spread across two lines by using `\`. The full code must be entered on one line, and the `\` character must be omitted. Note that the `user` option must be initialized via `debian-spamd` on Debian and Ubuntu, but via `spamd` on RHEL:

```
# in /etc/postfix/master.cf (Debian/Ubuntu)
smtp        inet  n  -   n  -  -  smtpd -o content_filter=spamassassin
spamassassin unix  -  n   n  -  -  pipe flags=R user=debian-spamd \
     argv=/usr/bin/spamc -f -e  /usr/sbin/sendmail -oi \
     -f ${sender} ${recipient}

# in /etc/postfix/master.cf (RHEL)
smtp        inet  n  -   n  -  -  smtpd -o content_filter=spamassassin
spamassassin unix  -  n   n  -  -  pipe flags=R user=spamd \
     argv=/usr/bin/spamc -f -e  /usr/sbin/sendmail -oi \
     -f ${sender} ${recipient}
```

Restarting Postfix makes the changes effective:

```
root# systemctl restart postfix
```

Test

To try out SpamAssassin, you can send a test message designed specifically for SpamAssassin to your server from an external email account. This message must contain the following string:

```
XJS*C4JDBQADN1.NSBN3*2IDNEN*GTUBE-STANDARD-ANTI-UBE-TEST-EMAIL*C.34X
```

Instead of typing the string, you can search Wikipedia for “GTUBE” (Generic Test for Unsolicited Bulk Email) and copy the string from there.

If you have a second Linux mail server available, you can install the incredibly handy `swaks` command there (Swiss Army Knife for SMTP) and send the test message as follows:

```
root# swaks --to hoover@a-company.com --server localhost \
      --data /usr/share/doc/spamassassin/examples/sample-spam.txt
```

The `sample-spam.txt` file is available at this location only on Debian and Ubuntu systems, provided that the `spamassassin` package is installed.

If everything works, the message will be detected as spam. The addressee receives the email marked as spam along with information about why it is probably spam. The text of the message should look similar to the following:

```
Spam detection software, running on the system "efd.a-company.com",
has identified this incoming email as possible spam.  The original
message has been attached to this so you can view it or label
similar future email.  If you have any questions, see
postmaster@a-company.com for details.
Content analysis details:   (999.8 points, 5.0 required)

 pts rule name          description
---- ------------------ ----------------------------------------
1000 GTUBE              BODY: Generic Test for Unsolicited Bulk Email
 ...
```

Whitelist

To generally accept email from a domain, you can whitelist the domain using the following statement:

```
# file /etc/[mail/]spamassassin/local.cf
...
whitelist_from *@friendly-company.com
```

Further options for formulating or automatically generating whitelists (e.g., on the basis of all addresses to which emails have been sent) are described at *https://wiki.apache.org/spamassassin/ManualWhitelist*.

26.7.1 Automatically Moving Spam to the Junk Folder

Marking emails as spam is helpful, but it is even more convenient if emails recognized as spam are automatically moved to a designated folder. This wish sounds trivial, but fulfilling it isn't easy. The following instructions are explicitly aimed at Debian/Ubuntu users.

LMTPD and Sieve

To move spam mails to a specific folder, you need two things: First, you need the *Local Mail Transfer Protocol Daemon* (LMTPD), Postfix extension that takes care of local mail delivery. Second, you need Sieve, a programming language for formulating mail-filtering rules:

```
root# apt install dovecot-lmtpd dovecot-managesieved
```

Sieve configuration

Mails detected as spam should be moved to the junk folder. The corresponding statements in the Sieve syntax must be stored in a `*.sieve` file. For files of this type, you should set up a new directory—for example, `/etc/dovecot/sieve-after`. There you can save the following text file:

```
# file /etc/dovecot/sieve-after/spam-to-folder.sieve
require ["fileinto","mailbox"];
if header :contains "X-Spam-Flag" "YES" {
 fileinto :create "Junk";
 stop;
}
```

This file must be compiled:

```
root# cd /etc/dovecot/sieve-after
root# sievec spam-to-folder.sieve
```

Dovecot configuration

To enable Dovecot to use both LMTPD and Sieve, a lot of changes are required in the configuration files. All the files mentioned ahead are located in `/etc/dovecot/conf.d`.

LMTPD expects mail usernames in the format `name@host.com`. So far in this chapter, however, I have assumed that only `name` is sufficient to identify a recipient within the mail server. For this reason, you need to set the `auth_username_format` parameter as follows:

```
# file /etc/dovecot/conf.d/10-auth.conf
...
auth_username_format = %Ln
...
```

To make the communication between Dovecot and LMTPD work, you need to comment out the existing `service lmtp` block in `10-master.conf` and replace it with the following lines:

```
# file /etc/dovecot/conf.d/10-master.conf
...
service lmtp {
  unix_listener /var/spool/postfix/private/dovecot-lmtp {
    mode = 0600
    user = postfix
    group = postfix
  }
}
...
```

To make the mail clients show the junk folder by default, add the `auto = subscribe` setting in the already existing `mailbox Junk` section:

```
# file /etc/dovecot/conf.d/15-mailboxes.conf
...
mailbox Junk {
  special_use = \Junk
  auto = subscribe
}
```

For Dovecot to use both LMTPD and Sieve, a valid postmaster address must be set in the `protocol lmtp` block and the `sieve` plugin must also be activated:

```
# file /etc/dovecot/conf.d/20-lmtp.conf
...
protocol lmtp {
  postmaster_address = postmaster@a-company.com
  mail_plugins = $mail_plugins sieve
}
```

Last but not least, you need to activate the Sieve rule you set up earlier. The following `sieve-after` setting takes into account all compiled Sieve files in `/etc/dovecot/sieve-after`:

```
# file /etc/dovecot/conf.d/90-sieve.conf
plugin {
  ...
  sieve_after = /etc/dovecot/sieve-after
}
```

Postfix configuration

The changes to `main.cf` are done quickly. The new line enables LMTPD for transporting mails to local directories:

```
# file /etc/postfix/main.cf
...
mailbox_transport = lmtp:unix:private/dovecot-lmtp
```

Test

To ensure that Postfix and Dovecot have detected all changes, you need to restart both programs:

```
root# systemctl restart postfix
root# systemctl restart dovecot
```

After that, you should send yourself another spam test mail using `swaks` (ideally from another computer):

```
root# swaks --to hoover@a-company.com --server localhost \
      --data /usr/share/doc/spamassassin/examples/sample-spam.txt
```

Ideally, the mail will now show up in the junk folder of your mail account. If not, you will have to start troubleshooting in the logging file, `/var/log/mail.log`.

RHEL

Basically, the procedure described here also works on RHEL. However, there are differences in detail, partly because the `dovecot-managesieve` package is not available. A tutorial for RHEL that uses the `dovecot-pigeonhole` package instead can be found at *https://www.linuxbabe.com/redhat/spamassassin-centos-rhel-block-email-spam*.

26.8 ClamAV (Virus Protection)

Unfortunately, spam is not the end of the story: If there are Windows computers on your network, you must also ensure that emails arrive free of viruses. Nowadays, email-borne viruses for Windows PCs are not the big issue they were a few years ago. However, email viruses are still a sporadic risk that must be minimized. It is therefore advisable to install another program on the email server that scans emails and their attachments for known viruses and deletes infected emails. Technically, an email filter works similarly to a spam filter, but attachments in particular must be scanned for patterns of known emails. This costs a lot of CPU time and is only effective if an up-to-date virus database is available.

ClamAV is the most popular open-source program for detecting viruses in files or emails. This primarily involves malware for Windows computers. Email viruses for Linux and macOS fortunately do not exist yet. Compared to commercial antivirus programs, ClamAV has rarely been a test winner in the past, but has usually done reasonably well. However, there is no guarantee that the very latest viruses will be detected correctly from square one.

Installation

Like SpamAssassin, ClamAV has several options for integration with Postfix. Here I present the Milter variant, which is the easiest to configure. Corresponding packages can be found in all common distributions. On Debian and Ubuntu, they are called `clamav`, `clamav-daemon`, and `clamav-milter`. The `freshclam` program is usually installed together with ClamAV. This program takes care of downloading the initial virus database and updating it regularly as a result. Take a look at the `/var/lib/clamav` directory (it must not be empty!) or read the `man` page for `freshclam`.

Configuration

To use ClamAV via the Postfix-Milter interface, you need to change some settings in `/etc/clamav/clamav-milter.conf`:

```
# in /etc/clamav/clamav-milter.conf
...
MilterSocket /var/spool/postfix/clamav/clamav-milter.ctl
MilterSocketGroup postfix
```

Then create the directory for the socket file so that both Postfix and ClamAV are allowed to read and write to it:

```
root# mkdir -p /var/spool/postfix/clamav/
root# chown clamav:postfix /var/spool/postfix/clamav/
root# chmod g+s /var/spool/postfix/clamav/
```

Restarting `clamav-daemon` and `clamav-milter` will ensure that ClamAV applies these changes:

```
root# systemctl restart clamav-daemon
root# systemctl restart clamav-milter
```

Now you need to modify `/etc/postfix/main.cf` so that Postfix redirects all incoming emails to ClamAV for checking. The paths of the socket files are relative to the postfix queue directory, `/var/spool/postfix`:

```
# addition to /etc/postfix/main.cf
...
smtpd_milters = unix:clamav/clamav-milter.ctl
```

Thanks to `postfix reload`, Postfix carries out the configuration change immediately:

```
root# systemctl reload postfix
```

Memory issues

ClamAV reads a database of malware signatures during the initialization. The program requires several GiB of memory. If you include the `ConcurrentDatabaseReload no` statement in `/etc/#clamav/clamav-milter.conf`, ClamAV is a bit more economical with memory, but it still required almost 2 GiB in my tests. If the available memory is insufficient, ClamAV will automatically be terminated by the Linux kernel. You can check this by using `systemctl`:

```
root# systemctl status clamav-daemon
clamav-daemon.service - Clam AntiVirus userspace daemon
Active: failed (Result: oom-kill) since Tue 2023-05-23 10:07:30 UTC;
25s ago
```

```
clamav-daemon.service: A process of this unit has been killed by the
OOM killer.
...
```

Even if ClamAV then uses slightly less memory in further operation, it does not make sense to use ClamAV in cloud instances with less than 4 GiB RAM.

ClamAV now inserts the following text in the header of each checked email: **Test**

```
X-Virus-Scanned: clamav-milter at a-company.com
X-Virus-Status: Clean
```

If ClamAV actually detects a virus, the email will not be redirected. Neither the sender nor the recipient is informed about this (`OnInfected Quarantine` setting). This procedure can be changed in `/etc/clamav/clamav-milter.conf`. The keywords used there are documented in `/usr/share/doc/clamav-milter/examples/clamav-milter.conf`.

To test the correct functioning of ClamAV, you should send a test message with the following text to your server from an external email account:

```
X5O!P%@AP[4\PZX54(P^)7CC)7}$EICAR-STANDARD-ANTIVIRUS-TEST-FILE!$H+H*
```

You don't have to type the text; you can also copy it from the following Wikipedia page: *https://en.wikipedia.org/wiki/EICAR_test_file*.

A message is then logged in the `/var/log/clamav/clamav.log` file or in the journal, which looks similar to the following listing:

```
root# journalctl -u clamav-daemon
...
/tmp/clamav-d01890e9f8302e614d9e822c2b73011.tmp (deleted):
  Eicar-Signature(2a54808d36676189bd1a0bea250ba8ba:1494) FOUND
```

26.9 SPF, DKIM, and DMARC

The three procedures discussed in this section store additional information about the behavior of your mail server in DNS records:

- **Sender Policy Framework**
  For SPF, a DNS record announces which hosts are authorized to send email for the domains.
- **DomainKeys Identified Mail**
  In the DKIM method, the public component of a key is stored in a DNS entry. The mail server signs the email with a private key. The recipient can use the public key to test the signature and thus establish beyond doubt that the email was really sent by the authorized host.

- **Domain-Based Message Authentication, Reporting, and Conformance**
  If both SPF and DKIM work on your mail server, you can also enable Domain-Based Message Authentication, Reporting, and Conformance (DMARC). In another DNS TXT entry, you thus publicly announce that your mail server supports SPF and DKIM and how you want to have it deal with mail that contradicts the SPF rules or that has an incorrect signature or no signature at all.

What is the point of all this? While SpamAssassin is intended to protect against unwanted spam, the task now is to configure one's own mail server to be as compliant with the rules as possible. Nothing is more annoying than when your own server's mails are classified as spam by Google, Apple, or Microsoft.

Now, no one can stop spammers from using SPF, DKIM, and DMARC as well. In this respect, DKIM and SPF are not a unique criterion for spam detection. But implementing these methods increases the chance that emails from your own server will not reach the recipient in the spam folder—at least a little; in my experience, the benefits are very limited. More generally, I get the feeling that mail giants like Microsoft like to consider all email that doesn't originate from their own network as spam, increasing the pressure on businesses to move to the cloud as well.

26.9.1 Sender Policy Framework

In SPF, you use a DNS entry of type `TXT` to specify the host or IP address from which mails of your own domain can be sent. Apart from setting up the correct DNS TXT entry, no further configuration work is required. However, the benefits are similarly negligible.

SPF syntax

The syntax is simple: The string must start with `v=spf1`. After that, `ip4:1.2.3.4` or `ip6:1234:...:6789` lists the IP addresses of the servers that are allowed to send emails for the mail server in question. In a simple mail server configuration without backup and relaying servers, these are simply the IP addresses of the server. The entry usually ends with `~all`. This means that all other IP addresses *should not* send emails for their own domain. Even stricter is the `-all` entry, which explicitly prohibits this. However, the `-all` setting is not recommended because it often causes problems with regard to redirecting emails.

The following TXT entry thus states that only emails sent from the IP addresses 95.217.156.167 or 2a01:4f9:c012:7e31::1 are to be considered valid emails from the `a-company.com` mail server:

```
user$ host -t TXT a-company.com
a-company.com descriptive text
  "v=spf1 ip4:95.217.156.167 ip6:2a01:4f9:c012:7e31::1 ~all"
```

The same information can be packaged even more simply:

```
user$ host -t TXT a-company.com
a-company.com descriptive text "v=spf1 a mx ~all"
```

This means that all IP addresses of host names specified by MX entries are valid. As you can see, this is consistent with the preceding example:

```
user$ host -t MX a-company.com
a-company.com mail is handled by 10 efd.a-company.com.

user$ host -t A efd.a-company.com
host -t A mail.a-company.com
efd.a-company.com has address 95.217.156.167

user$ host -t AAAA efd.a-company.com
efd.a-company.com has IPv6 address 2a01:4f9:c012:7e31::1
```

It's especially easy to determine the SPF entry that is appropriate for your mail server using a form on the following page: *https://www.spf-record.com/generator*.

TXT versus SPF entry

The original plan was to define a separate type of DNS record for SPF records. However, it turned out to be easier to simply place the information in the already provided text entries. You can find more information on this here:

- *https://en.wikipedia.org/wiki/Sender_Policy_Framework*
- *https://www.spfwizard.net/index.php?mod=whatis*

So the only question that remains is: How do you change the DNS configuration? Competent providers or domain registrars give you the option of changing the TXT entries during DNS configuration via a web interface (see Figure 26.2 in Section 26.1).

26.9.2 DomainKeys Identified Mail

DKIM is considerably more complex and time-consuming to implement. When a DKIM-configured mail server sends a mail, it uses a private key stored on the mail server and adds a signature to the mail.

The mail server of the recipient can now read the DNS TXT entry of the sender for checking purposes. This contains the public component of the

key, which can be used to check whether the email and the signature match (see Figure 26.4). DKIM can therefore be used to check whether the email actually originates from the sender address specified in it—and that counts for a lot. Only the legitimate mail server has the private key and can thus perform a correct signature.

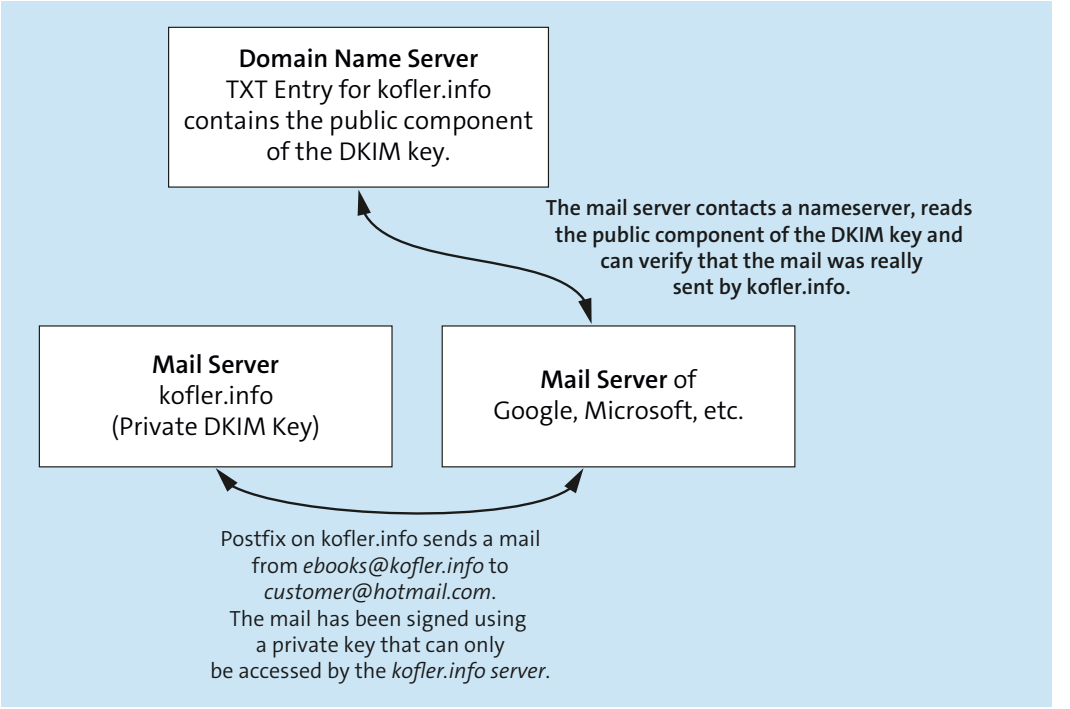


Figure 26.4 Conceptualization of DomainKeys Identified Mail

26.9.3 OpenDKIM

Installing and setting up OpenDKIM

OpenDKIM is an open-source implementation of DKIM. The program can both sign outgoing mails with its own key and verify the signature of incoming mails. OpenDKIM is compatible with various MTAs. This following instructions demonstrate the interaction of DKIM with Postfix, with the main focus on correct signing.

On Debian/Ubuntu, you want to install the `opendkim` and `opendkim-tools` packages. On RHEL, only `opendkim` is sufficient, but you must first activate the CRB and EPEL package sources. Then you need to set up a directory for the DKIM keys:

```
root# mkdir -p /etc/opendkim/keys
root# chown -R opendkim:opendkim /etc/opendkim
root# chmod go-rw /etc/opendkim/keys
```

opendkim.conf

The central configuration file of OpenDKIM is `/etc/opendkim.conf`. You can set up this file like the following pattern. Details about the `trusted`, `key.table`, and `signing.table` files referenced by the configuration follow shortly. You can find details on the various `opendkim.conf` options with `man opendkim.conf`:

```
# file /etc/opendkim.conf
# OpenDKIM acts as a mail filter (= milter) in the
# signer (s) and verifier (v) modes and uses
# a port or socket file for communication
Mode                    sv
# Socket                local:/var/run/opendkim/opendkim.sock
# Socket                inet:12345@localhost
Socket                  inet:8891@localhost

# OpenDKIM uses this user or group
UserID                  opendkim:opendkim
UMask                   002
PidFile                 /var/run/opendkim/opendkim.pid

# Restart OpenDKIM in case of problems, but max. 10 times per hour
AutoRestart             yes
AutoRestartRate         10/1h

# Logging (if everything works, possibly reduce)
Syslog                  yes
SyslogSuccess           yes
LogWhy                  yes

# Method how header and body are to be processed
# by OpenDKIM
Canonicalization        relaxed/simple

# internal mails (sign, do not verify)
InternalHosts           refile:/etc/opendkim/trusted
# hosts that are trusted (avoids logging warnings)
ExternalIgnoreList      refile:/etc/opendkim/trusted

# Which encryption keys are to be used for which
# domains?
# (refile: for files with regular expressions)
SigningTable            refile:/etc/opendkim/signing.table
KeyTable                /etc/opendkim/key.table
```

```
# Use this signature algorithm
SignatureAlgorithm      rsa-sha256

# Sign sender (From header)? This prevents manipulated
# headers between signer and verifier. In Debian the sender is
# signed by default because it is often used as an identity key
# and is therefore safety critical.
OversignHeaders         From

# Optional for DNSSEC. On Debian/Ubuntu, root.key
# is part of the dns-root-data package.
TrustAnchorFile         /usr/share/dns/root.key
```

Note that on Debian and Ubuntu, the `/etc/default/opendkim` file is also analyzed when OpenDKIM is started. Settings made there take precedence over `opendkim.conf`. I assume in these instructions that you disable the default settings provided there by inserting comment characters. (Only `RUNDIR` should be left as it is.)

RHEL also has its own peculiarities: Its sample file `opendkim.conf` provides port 8891 as an option. The corresponding line is commented out. By default, `opendkim` uses a socket file. Now you might think that any other free port would work just as well, but that is not the case: there is only a SELinux exception rule for port 8891 that allows `opendkim` to be used. If you want to use a different port, you must run the following command, specifying the desired port number instead of `17654`, of course:

```
root# semanage port -a -t milter_port_t -p tcp 17654
```

No signature for internal mails

In my example, I assume that the other configuration files are located in the `/etc/opendkim` directory. On Debian and Ubuntu, you must first create this directory using `mkdir`.

The `trusted` file specifies which hosts the mail server trusts—that is, for which hosts the DKIM signature is waived. Replace the two `a-company.com` entries in the following with corresponding entries for the host name of your mail server:

```
# file /etc/opendkim/trusted
127.0.0.1
::1
localhost
a-company.com
efd.a-company.com
```

Determine keys for host names

Next, use an editor to set up the `signing.table` and `key.table` files according to the following pattern. `signing.table` specifies which key is to be used for which `From` addresses. The second column in `signing.table` refers to the first column in `key.table`. Of course, you need to replace `a-company.com` and `efd` with your own names:

```
# file /etc/opendkim/signing.table
# for emails from xxx@a-company.com use the
# 'efd' key for signing
*@a-company.com efd
```

`key.table` then specifies the actual location of the key files. The second column consists of three parts: the host name (here `a-company.com`), the selector (here `202305`) and the actual file name. Using a time specification for the selector helps to keep track of when the keys were generated—in this case, in May 2023:

```
# file /etc/opendkim/key.table
# The 'efd' key is located in
# file /etc/opendkim/keys/efd.private
efd  a-company.com:202305:/etc/opendkim/keys/efd.private
```

If your mail server is responsible for multiple hosts, you must add corresponding entries to `key.table` and `signing.table` for all additional hosts.

Generating a key

The `opendkim-genkey` command generates a private key (`name.private`) and a public key (`name.txt`). The private key remains on the server and must not fall into the wrong hands! The public key is later stored in the DNS TXT entry of the mail server. The main options of the command are as follows:

- `-d domain` specifies the domain for which the keys should apply. The option is currently only used to add a comment to `name.txt`. Without the option, `opendkim-genkey` uses the `example.com` string.
- `-b 2048` specifies that a key of length 2,048 bits is to be generated. Larger keys are currently incompatible with the current mail server infrastructure. By default, `opendkim-genkey` uses 1,024 bits, but this is often no longer considered sufficiently secure.
- `-r` specifies that the key may only be used to sign emails (*restricted*).
- `-s selector` specifies a selector string to identify the key pair (by `default`). The selector string determines the names of the key files. So `-s 202305` causes the `202305.txt` and `202305.private` files to be created. *Caution:* Any existing key files will be overwritten without confirmation.

To generate keys that match the preceding example, you need to run the following command:

```
root# cd /etc/opendkim
root# opendkim-genkey -d a-company.com -b 2048 -r -s 202305
```

The files must now be placed according to the information in `key.table`. Finally, you want to make sure that only the `opendkim` user is allowed to access the files and directories:

```
root# cd /etc/opendkim
root# mkdir keys
root# mv 202305.private keys/efd.private
root# mv 202305.txt     keys/efd.txt
root# chown -R opendkim:opendkim /etc/opendkim
root# chmod -R go-rwx /etc/opendkim/keys
```

Adding a DNS TXT entry

The next step is to save the public component of the DKIM key—that is, the content of `efd.txt` in the logic of this example—in a TXT entry in the DNS configuration of your host:

```
root# cat /etc/opendkim/keys/efd.txt
202305._domainkey IN TXT
  ( "v=DKIM1; h=sha256; k=rsa; s=email; "
    "p=MIIBIjA..."
    "zLVkXtu6v..." )  ; ----- DKIM key 202305 for a-company.com
```

For this purpose, you must use the character string before the `IN` keyword for the **Name** or **Hostname** field of the DNS entry and the entire text between the parentheses for the **Target** field (see Figure 26.5).

It may take several hours for the DNS data change to become generally available on the internet. You can read the TXT entries of a host using the `host -t TXT` command, prefixing your domain name with `<selector>._domainkey`, here `202305._domainkey`:

```
root# host -t TXT 202305._domainkey.a-company.com
202104._domainkey.a-company.com descriptive text
  "v=DKIM1; h=sha256; k=rsa; s=email; " "p=
MIIBIjANB..." "zLVkXtu6vo9p1..."
```

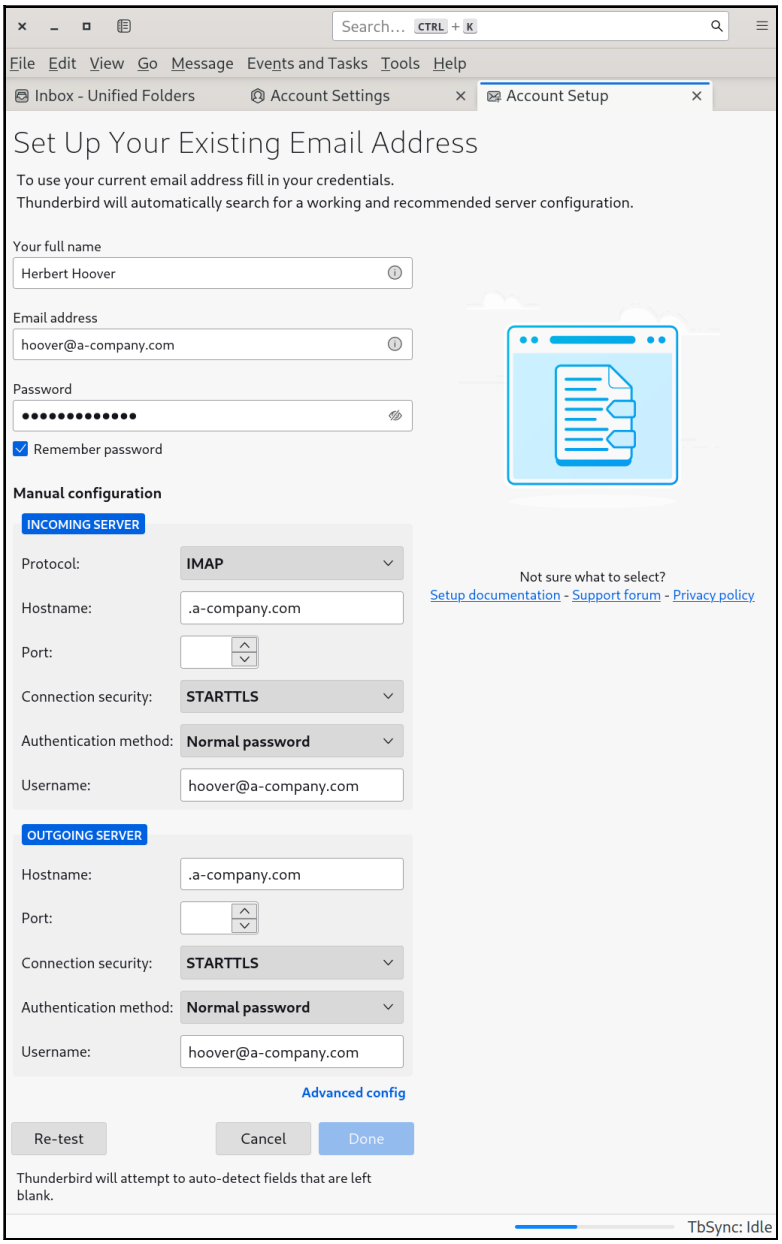


Figure 26.5 Definition of New TXT Entry in Web Interface of Hosting Company

The changed settings require a restart of OpenDKIM:

Testing OpenDKIM

```
root# systemctl restart opendkim
```

As soon as the public key stored on the name server is available (you test this using the `host -t TXT` command, shown previously), you can use the

opendkim-testkey command to check locally whether everything has been set up correctly. Provided that the command does not return any error messages, everything is fine. The -vvv option causes the command to provide at least some debugging data:

```
root# opendkim-testkey -d a-company.com -s 202305 -vvv
  opendkim-testkey: using default configfile /etc/opendkim.conf
  opendkim-testkey: checking key '202305._domainkey.a-company.com'
  opendkim-testkey: key not secure
  opendkim-testkey: key OK
```

Key not secure indicates that DNSSEC is not used. But this has nothing to do with DKIM. If, on the other hand, the *record not found* message appears, OpenDKIM cannot find the required key—and then there really is a configuration problem.

26.9.4 Postfix Configuration for OpenDKIM

For Postfix to sign outgoing emails with OpenDKIM, OpenDKIM must be set up as a mail filter (*milter*). For this purpose, the smtpd_milters and non_smtpd_milters parameters are provided in the Postfix configuration. If OpenDKIM is the only milter, the correct configuration looks as follows:

```
# in /etc/postfix/main.cf
milter_default_action = accept
milter_protocol   = 6
smtpd_milters     = inet:localhost:8891
non_smtpd_milters = inet:localhost:8891
```

It's possible that there was already a milter—in which case, the OpenDKIM milter (separated by a comma) is simply added.

Some notes on the parameters:

- milter_default_action = accept means that Postfix should accept mail even if a milter is faulty, not responding, misconfigured, and so on.
- Many instructions talk about milter_protocol = 2. However, this only applies to old Postfix versions (prior to version 2.6). Current Postfix versions default to milter_protocol = 6. So the configuration works even if you omit the setting altogether.
- You may wonder what the difference is between smtpd_milters and non_smtpd_milters. smtpd_milters applies to emails delivered via the SMTP protocol, while non_smtpd_milters, in turn, applies to local emails generated and sent by sendmail or by a PHP script, for example.

Test

To test DKIM in the interaction of Postfix and OpenDKIM, you should first send yourself a mail and then take a look at the header. This should contain a line starting with the DKIM signature—for example, like this:

```
DKIM-Signature: v=1; a=rsa-sha256; c=relaxed/simple; d=a-company.com;
    s=202305; t=1684851307;
    bh=g3zLYH4xKxcPrHOD18z9YfpQcnk/GaJedfustWU5uGs=;
    h=Date:From:To:Subject:From;
    b=
fHtO5qGkWzRYXrLHxCwldlkSTzHWLYWTQOCACa1R6qGbBd1txLmbBknIfV7FigOnt...
```

Now send an email from your own mail server to a mail server that you know supports and uses DKIM. A good choice is Google (i.e., Gmail). In the Gmail web interface, open the email and run **View Original** (see Figure 26.6).

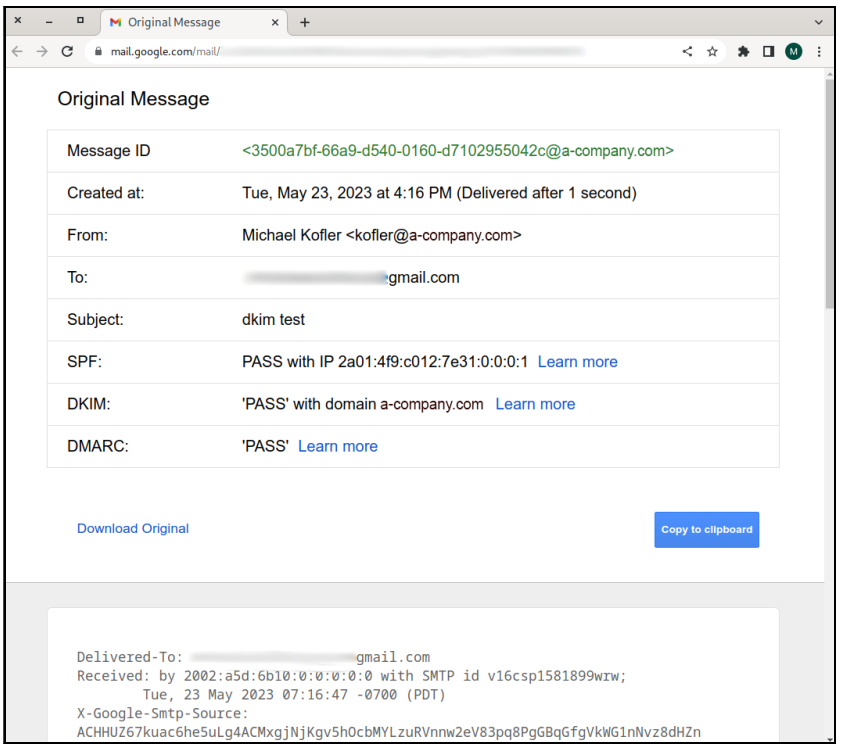


Figure 26.6 Checking DKIM Configuration in Gmail

26.9.5 Domain-Based Message Authentication, Reporting, and Conformance

If both SPF and DKIM are working on your mail server, you can still consider enabling DMARC. To do this, you must use another DNS-TXT entry to publicly announce both that your mail server supports SPF and DKIM and

how to deal with mail that contains your host name in the sender address but was sent from other mail servers or is not correctly signed with DKIM.

Setting up DMARC doesn't require much work. All you need to do is define another DNS TXT entry that uses `_dmarc` as the host name and has an entry according to the DMARC syntax. You can use the DMARC settings of other mail providers as a guide:

```
user$ host -t TXT _dmarc.yahoo.com
  _dmarc.yahoo.com descriptive text
  "v=DMARC1; p=reject; pct=100; rua=mailto:d@rua.agari.com; ruf=...;"

user$ host -t TXT _dmarc.gmail.com
  _dmarc.gmail.com descriptive text
  "v=DMARC1; p=none; sp=quarantine; rua=mailto:mailauth-
reports@google.com"

user$ host -t TXT _dmarc.hotmail.com
  _dmarc.hotmail.com descriptive text
   "v=DMARC1; p=none; rua=mailto:d@rua.agari.com;
    ruf=mailto:d@ruf.agari.com;fo=1:s:d"
```

The most radical is Yahoo. (This company originally developed DMARC.) Mails that do not comply with SPF and DKIM should simply be discarded. Error reports can be sent to `dmarc_y_rua@yahoo.com`.

Gmail and Hotmail take a much more relaxed view of the matter: `p=none` suggests accepting mails that are not correctly signed anyway (*monitor mode*). But these mail providers are also interested in error messages and provide corresponding mail addresses.

Like all other techniques, DMARC is controversial. If DMARC is run strictly, as it's with Yahoo, then mail redirected through mailing lists often causes problems. This is because most mailing list programs do not modify the *From* line, which causes a discrepancy between the specified sender and the mail server of the mailing list when forwarding.

So if you want to set up a DMARC record, I think you should take Gmail as a model and enable DMARC without the reject rule in monitoring mode only. You can do without `rua` if you are not interested in automated feedback mails from other server operators:

```
user$ host -t TXT _dmarc.a-company.com
  _dmarc.a-company.com descriptive text "v=DMARC1; p=none"
```

If you specify a feedback address using the `rua` keyword, you will receive regular feedback (usually daily) in the form of a compressed XML file from

some mail providers that use DKIM and to which your server sends mail. The file shows whether DKIM has been used correctly and whether (probably without your knowledge) other mail servers have tried to send mail on your behalf, triggering DKIM errors.

26.10 Configuration Test and Troubleshooting

Spam test

When you visit *http://isnotspam.com* or *https://www.mail-tester.com*, you will receive a random mail address to which you can send a test mail. However, the mail should contain real content, not just "test"; otherwise, it will fail the spam test. The email is then checked, and you receive a report that checks various aspects of the configuration (see Figure 26.7).

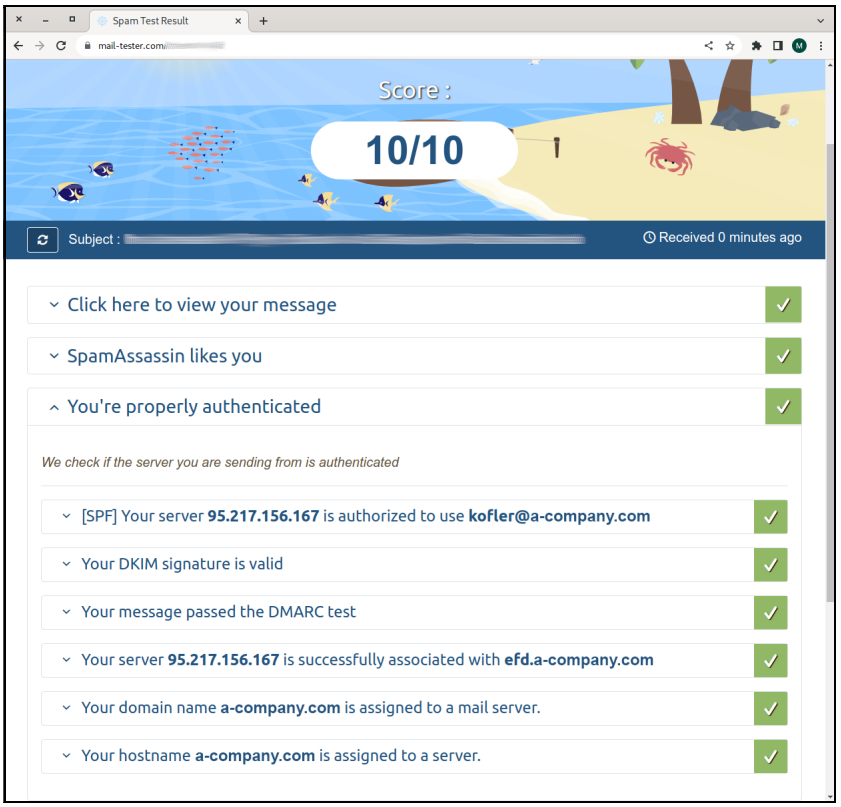


Figure 26.7 Analysis of Email Sent to mail-tester.com

mxtoolbox.com

At *https://mxtoolbox.com*, you can specify the host name of your mail server. The page checks if the DNS configuration is correct and if your server is blacklisted.

checktls.com

On *https://www.checktls.com*, you can specify a valid email address of your server. Scripts of the page check if the TLS configuration is correct.

Troubleshooting using telnet

There are a number of ways to check the configuration of your mail server. Surprisingly, if there are problems with SMTP authentication, the outdated `telnet` program is a valuable troubleshooting tool. You can run this command on any computer. It does not have to be the mail server.

The connection setup via port 25 starts with text messages. You must reply to the server's welcome message with `EHLO <hostname>`. Postfix then announces that further communication may be encrypted according to the STARTTLS method and that authentication may be done by a password in plain text. You can use `quit` to end the "conversation" with the mail server:

```
user$ telnet mail.a-company.com 25
Trying 2a01:4f9:c012:7e31::1...

Connected to efd.a-company.com.

Escape character is '^]'.
220 efd.a-company.com ESMTP Postfix (Ubuntu)
EHLO a-company.com
250-a-company.com
...
250-STARTTLS
250-AUTH PLAIN LOGIN
...
quit
Connection closed by foreign host.
```

Chapter 27
Samba

Samba allows you to place files or directories on the local network. Users with Windows, Linux, and Apple computers can access it. A finely differentiated authentication and rights system controls who can read or modify which files. Samba is thus the central node for data exchange on a local network—be it in a company, in an organization, or at home.

The name *Samba* is derived from the abbreviation *SMB* for the *Server Message Block* protocol. The first SMB concepts were developed by IBM. Later the protocol was extended by several companies, Microsoft in particular.

Samba is often regarded as a linking element between the Linux and Windows worlds. But that falls short of the mark. SMB is often used even in Linux-only environments because it's very easy to use (provided all components are configured correctly). The GNOME and KDE file managers support SMB by default, SMB directories can be mounted directly in the Linux directory tree thanks to CIFS, and so on.

If you're looking for a Unix-type alternative without a Microsoft background, NFS is the most likely choice. However, the use of NFS requires uniform login names on all client computers or the use of LDAP. Furthermore, NFS is incompatible with the Windows world.

This chapter describes the basic functions of Samba 4 from the server perspective. **Links**

If you want to use Samba's numerous advanced features, like authentication via LDAP or the use of Samba as a primary domain controller (PDC), refer to the available literature. A lot of information about Samba can also be found on the following websites:

- *https://wiki.samba.org*
- *https://help.ubuntu.com/community/Samba*
- *https://wiki.archlinux.org/title/Samba*

27.1 Basic Principles and Terminology

Before you begin configuring a Samba server, you should understand the underlying concepts and associated terminology. This section is designed to help you do that.

NetBIOS In its original form, SMB is based on the *Network Basic Input/Output System* (NetBIOS) protocol, originally developed by IBM. However, NetBIOS has been repeatedly updated. NetBIOS primarily consists of three services:

- **Name service**
  This method for exchanging computer names is comparable to DNS on Unix/Linux. The names can be managed either centrally by a NetBIOS name server (NBNS) or decentrally. In the latter case, each client sends a message to all other clients on the network at computer startup, telling them under which name it is present.
- **Session service**
  Similar to TCP, this communication mechanism enables an orderly exchange of data between two computers in the form of packets. During this process, the integrity is checked, and faulty or lost packets are requested again.
- **Datagram service**
  This variant of the session service does not check whether the data arrives correctly. However, the datagram service has the advantage that data can be sent to several computers at the same time.

WINS On Windows, the NBNS is implemented by the Windows Internet Name Service (WINS). WINS servers can be set up with Samba or with current Windows versions. WINS is optional in current Windows and Samba versions, especially if there is already a name server running on the local network. If Samba is used in combination with IPv6, a name server is even mandatory instead of WINS. On Linux, you can use the `dnsmasq` program for this purpose.

Browsing How does a Windows client know which other computers are on the network? In the past, the answer was *browsing*. This term refers to the management of computers located on the network via the SMB protocol in version 1.

WS-Discovery Today, SMB1 is hardly used anymore due to security issues. In current versions, the browsing functions have been removed from SMB. Instead, the WS-Discovery (WSD) protocol is used on Windows. Although this protocol has been in use for almost 15 years now, it still causes many problems in the interaction between Windows and Linux:

- Thanks to Avahi (see Chapter 15, Section 15.5), Linux recognizes Samba servers, Apple computers, and NAS devices on the local network, but unfortunately fails with directories offered by Windows computers on the local network.

If you use older distributions, access to network directories often succeeds only after a trip to the terminal (see Section 27.7). This is anything but user-friendly.

- Conversely, Windows computers sometimes do not recognize active Samba servers on the local network. On the Linux side, this can be solved by using a current Samba version with SMB3 and installing `wsdd` or `wsdd2`, and on the Windows side by activating the discovery functions if necessary. Details on this also follow in Section 27.7.

SMB versions

There are several versions of the SMB protocol. Table 27.1 summarizes which protocol version is supported in which Windows and Samba version. The mapping between SMB and Samba versions is difficult because new SMB versions of Samba are supported step by step. For example, some SMB 2.1 functions have already been implemented in Samba 3.5.

SMB	Windows	Samba
SMB 1.0	Older Windows versions	Older Samba versions
SMB 2.0	Windows Vista (2007)	As of Samba 3.6
SMB 2.1	Windows 7 (2009)	As of Samba 3.6
SMB 3	Windows 8 (2012)	As of Samba 4.1
SMB 3.1	Windows 10 (2015)	As of Samba 4.3

Table 27.1 SMB Versions

Basically, both Windows and Samba are backward compatible, so that the interaction of different Windows and Samba versions usually succeeds without any problems. Only SMB1 is now blocked by most installations for security reasons.

27.1.1 Access Rights and Security Systems

The term *shares* refers to directories or printers that are made available to other computers. There are various types of access control that regulate who can access which shares:

- **Share-level security**
  In the simplest type of access control, each directory and each printer is given its own password. This process is hardly ever used anymore. The obvious disadvantage is the large number of passwords required: if three objects are shared on each of 10 computers, this already results in 30 passwords. In larger networks, that inevitably produces chaotic conditions.

- **Workgroup security**
  This extension of share-level security allows mutual access to objects only if the computers belong to the same workgroup. This does not yet significantly improve security: each computer can declare itself to belong to any workgroup without central administration. Share-level security thus operates according to a decentralized peer-to-peer process, as opposed to the increasingly centralized client/server processes for user-level and domain-level.
- **User-level security**
  User-level security on the client side requires the user to log in with a name and password. If the user wants to use any data of a Samba server or another Windows PC, their current name and password are considered as access authorization. In addition, both computers must belong to the same workgroup.

  Each network object is therefore not simply assigned a password. Instead, it is linked to a user or to a list of named users or to all users in a group. When user-level security is implemented with Samba, a separate database of user names, group memberships, and passwords is required.

  Let's look at an example: User X works on computer A. For X to retrieve data from Samba server S, X must be registered as a user on both A and S, each with the same name and password. Now computer A breaks down. X switches to computer B. To be able to access their data on S, user X must also be created on B (again with a password).

  If X decides to change the password, this change must be performed on server S and subsequently on each client (A, B, etc.). Decentralized password management and authentication is thus an inherent problem with this concept.

  On the other hand, user-level security is straightforward to manage in small networks: If you want to put a photo directory on the local network that is accessible to everyone, you can set up a new user (`photos`) and a password for it and share this data with everyone who should have access. The first access to the directory is attempted by the file manager with the current login data of the respective user. This does not work. Then a login dialog for the directory appears. The user enters `photos` and the associated password and is allowed access! Usually, the file manager remembers the login data so that everything works like magic when you access it later.
- **Domain-level security**
  With Windows NT 4, Microsoft introduced so-called domains into its network world. The concept of access management is quite similar to

user-level security. The differences concern the way in which the user database is managed and how authentication is carried out.

The clients access the user database managed centrally by the server when logging in. Access rights are managed by a kind of login token. The client receives access information at login, which is valid throughout the network until logout. Although this difference is not visible to the user, it represents a fundamental difference in internal administration and is much more efficient for the server to handle.

Domain-level security requires that there be a primary domain controller (PDC) on the network. In larger networks, the PDC can be assisted by a few backup domain controllers (BDCs) so that everything doesn't come to a standstill just because the PDC has just failed.

- **Active Directories**
  To simplify the management of large networks, Microsoft has extended domain-level security to include Active Directories. Lightweight Directory Access Protocol (LDAP) is used for authentication. Active Directories can be used to organize the network and its domains hierarchically. Furthermore, the administration of the computer names is done by a name server, like on Linux.

Samba has long understood all five of these security systems on the client side. Samba has been able to assume the role of an Active Directory domain server since version 4. However, this book only covers the user-level security system. In this context, we also speak of a Samba configuration as a standalone server. If you want to use Samba as a PDC or as an Active Directory server, you will need further literature.

27.1.2 Centralized or Decentralized Server Topology?

Aside from the security system, there are two fundamentally different strategies for exchanging data between computers on a local network via Samba:

- **Central topology**
  A central Samba server makes network directories available to all users. It is the administrator's task to set the access rights of the individual directories so that there are both private directories for individual users and more or less public directories for data exchange in user groups. Instead of a self-configured Windows or Linux computer, the server can simply be a NAS device. All commercially available NAS devices use Samba internally.

The advantages of this configuration are that all data can be backed up centrally on the server and that individual users do not have to worry about setting up network directories themselves. Of course, there are also disadvantages: the system is relatively inflexible, and any change must be made by an administrator. Moreover, the consequences of a server failure are fatal for the entire network.

- **Decentralized topology**
  In this case, each computer that wants to make data available to other users on the network makes it available itself. Both Windows and Linux (strictly speaking, GNOME or KDE) support the user with sharing dialogs.

  The advantage of this configuration variant is the decentralized approach, where everyone is responsible for themselves and no administrator is required. As the network size increases, the configuration naturally becomes confusing. In addition, centralized backups are nearly impossible.

A centralized topology is more appropriate for an enterprise deployment. However, if it's only a matter of quickly copying a few files from one computer to the next in a private environment, an ad hoc configuration through the corresponding KDE or GNOME dialog is of course sufficient. The main problem is that the configuration dialogs provided by KDE or GNOME are almost as half-baked today as they were 10 years ago.

27.1.3 NAS Instead of Tinkering with Samba?

After this introduction, it's probably clear to you that the manual administration of Samba, which I will explain on the following pages, is not quite trivial. A simple and popular alternative is to purchase a network-attached storage (NAS) device and configure it via a web interface. This is not trivial either, but definitely less tedious than changing `smb.conf` manually.

Another alternative is to install NAS software on your PC or on a Raspberry Pi. This turns your computer into a NAS device that provides similar functions and operation comparable to a commercial NAS device. The main advantage is that you can continue to use existing hardware.

openmediavault

As to the software, I recommend the *openmediavault* (OMV for short) open-source program for your own NAS projects. The standard variant is based on Debian and can be downloaded as an ISO image similar to this and installed on a PC.

The procedure is slightly different if you want to implement the project with a Raspberry Pi: there you first install Raspberry Pi OS Lite and then run a setup script that installs and sets up all the necessary additional

components. Note that the Raspberry Pi is only suitable for NAS use from model 4B on. On older models, the ethernet and USB interfaces are simply too slow.

27.2 Basic Configuration and Commissioning

Installation

Many distributions have separate packages for the client and server applications. The client packages are usually installed by default so that access to network directories works right away. If you want to share network directories yourself, you also need the server functions, which are contained in the `samba` package in most distributions.

Start

Samba provides its services through two background processes:

- `nmbd` is used for internal administration and as a name server. The daemon also takes care of the browsing functions (SMB1 only) or can act as a WINS server.
- `smbd` provides the interface for the clients and gives them access to the directories and printers.

In current distributions, the processes are started by systemd. The service names vary depending on the distribution. For example, Debian and Ubuntu use `smbd` and `nmbd`, while Fedora, RHEL, and SUSE use `smb` and `nmb`. If you configure Samba as a domain controller for Active Directories, you must also start the `samba-ad-dc` service. However, I will not go into that in this chapter.

Remember that system services on Fedora, RHEL and SUSE are not started automatically after the package has been installed. Thus, you must run `systemctl enable --now <name>` for each service (see Chapter 10, Section 10.5.2).

Samba version

You can use `smbstatus` or `sbmd -V` to easily test which Samba version is running on your computer:

```
root# smbstatus
Samba version 4.18.2 ...
```

smb.conf

The central configuration file for Samba is `/etc/samba/smb.conf`. The file is composed of a `[global]` section for the basic settings and any number of additional sections for sharing directories, printers, and so on. Each section is introduced by `[resource name]`. Comments can begin with the `;` or `#` character. However, you cannot add a comment following a parameter setting. Comments therefore always take up an entire line.

The following lines show the global section of the default Samba configuration on Ubuntu in slightly shortened form. In other distributions, the file is sometimes even shorter, because there is no need to explicitly repeat default settings. The [printers] and [print$] sections, which allow access to printers and printer drivers, are not printed here:

```
# file /etc/samba/smb.conf bei Ubuntu
[global]
   workgroup = WORKGROUP
   server string = %h server (Samba, Ubuntu)
   log file = /var/log/samba/log.%m
   max log size = 1000
   logging = file
   panic action = /usr/share/samba/panic-action %d
   server role = standalone server
   obey pam restrictions = yes
   unix password sync = yes
   passwd program = /usr/bin/passwd %u
   passwd chat = ...
   pam password change = yes
   map to guest = bad user
   usershare allow guests = yes
```

On Fedora and RHEL, smb.cnf has a very similar structure. However, some options are missing that merely document the default state of Samba. Instead, some lines are included to share all home directories as network directories. In addition, the workgroup name is SAMBA instead of WORKGROUP.

Removing comments from smb.conf

In some distributions, smb.conf is also used for documentation. The resulting configuration file is then endlessly long and confusing. But the following two commands can help you in this context:

```
root# cd /etc/samba
root# cp smb.conf smb.conf.orig
root# egrep -v '^#|^;|^$' smb.conf.orig > smb.conf
```

Server identification

workgroup allows you to set the name of the workgroup. This is probably the first setting you will change—to set the name of your own workgroup within which Samba should operate. server string specifies the name under which the server identifies itself. %h is replaced by the host name.

Logging

The log file, max log size and logging parameters control which data is logged by Samba and where. If Samba crashes, the panic-action script is executed. It sends an email to root containing information about the error that occurred. panic-action has no effect if there is no email system installed on the server.

Security model

Due to the `server role = standalone server` setting, the user-level security model applies. Alternatively, the `security = user` setting would also activate this security model. Even if both settings are missing, user-level security is the default setting in current Samba versions.

Passwords

The `unix password sync`, `passwd chat`, and `pam password change` keywords describe whether and how Samba should synchronize its passwords with the Linux passwords. By default, passwords are stored internally in Samba in the relatively simple TDB database system (default setting: `passdb backend = tdbsam`). Details on managing Samba passwords follow in Section 27.3.

Guests

`map to guest` and `usershare allow guests` control how Samba handles unauthenticated users—that is, users who log in with an invalid name or password. The meanings of these and some other `guest` parameters are discussed in Section 27.4.

Samba standards

On Debian and Ubuntu, `smb.conf` contains some statements that are actually not needed. For example, the `obey pam restrictions = yes` setting only affects password management if passwords are not encrypted. Because this is the case by default, the setting will be ignored.

In the further course of this chapter, you will get to know a lot more Samba parameters, but of course far from all of them. Detailed information on all setting options can be found in `man smb.conf`.

27.2.1 Configuration Changes and Status

For changes to `smb.conf` to take effect, you must prompt Samba to reload the configuration files:

```
root# systemctl reload smb[d]
root# systemctl reload nmb[d]
```

testparm

If you want to make major changes to `smb.conf`, you should first check the file for syntactical errors using `testparm`:

```
root# testparm
Load smb config files from /etc/samba/smb.conf
Processing section "[printers]"
Processing section "[print$]"
Loaded services file OK.

Server role: ROLE_STANDALONE
Press enter to see a dump of your service definitions <Return>
[global]
   server string = %h server (Samba, Ubuntu)
   ...
```

If you run `testparm` with the `-s` and `-v` options, the command returns a long list of all `smb.conf` options without prompting. This is useful if you want to find out which settings apply by default—that is, for options that you have not explicitly set yourself.

You can determine the current state of the Samba server by using `smbstatus`. The command also provides a list of all currently active connections.

27.2.2 Firewall

On Fedora, SUSE, and RHEL, the default active firewall blocks Samba. To be able to use the network services, you must enable the Samba- or Windows-specific TCP ports 135, 139, and 445 as well as UDP ports 137, 138, and 445 for the interface to the local network—both on the server and on the client computers.

For Fedora, RHEL, and SUSE, `firewall-cmd` is the command of choice. You can either change the zone of the network interface through which the Samba data flows, or you can keep the zone and define exception rules for Samba. The latter method is outlined here:

```
root# firewall-cmd --get-active-zones
public
  interfaces: enp1s0
root# firewall-cmd --permanent --zone=public \
  --add-service=samba
root# firewall-cmd --permanent --zone=public \
  --add-service=samba-client
root# firewall-cmd --reload
```

Opening the SMB ports within the local network is essential to be able to use Samba. However, general deactivation of the firewall is not recommended! In addition, the entire local network should always be protected from the internet by another firewall, which runs on the Wi-Fi router, for example. If this is not the case, the shared directories are accessible to the whole world or are only secured by the passwords of the Samba user management. You can find background information about firewalls in Chapter 29.

27.2.3 Securing Samba

When configuring Samba, you should generally try to restrict access to the network directories as much as possible. In this section, I'll show you some ways to do that.

interfaces

You can use `interfaces` to specify which network interfaces Samba is allowed to communicate over. The specification of the interfaces is not done via the names of the interfaces but via the address range used by these interfaces. You can also specify multiple ranges, simply separating them with spaces. Don't forget `localhost`; otherwise, administration tools like `smbclient` won't work on the server!

The `interfaces` option is relevant if there are multiple network interfaces on your computer. On many computers, there are interfaces not only to physical network adapters, but also to the virtual network adapters of various virtualization programs! By default, Samba serves *all* network interfaces.

The settings implemented by `interfaces` only take effect if you also enable the `bind interfaces only` option, as in the following listing:

```
# /etc/samba/smb.cnf
[global]
   ...
   # Samba should only serve the local network 10.22.*.*
   bind interfaces only = yes
   interfaces           = 10.22.0.0/16 127.0.0.1
```

Hosts

Furthermore, you can use `hosts allow` to explicitly list those hosts that are allowed to communicate with Samba. The host names, IP addresses, or IP address ranges must be separated by spaces. `hosts allow` allows an even more precise selection than `interfaces`. In addition, you can use `hosts deny` to prohibit individual hosts or addresses from using Samba:

```
# /etc/samba/smb.cnf
[global]
   hosts allow          = clientA clientB clientC
```

IPv4 and IPv6

In principle, Samba communicates via both IPv4 and IPv6. In a dual-stack configuration on the LAN, it's sometimes advisable for security reasons for Samba only to "speak" IPv4. For this purpose, you simply need to enter only IPv4 addresses or address ranges for `interfaces` and `host`. In such cases, you should avoid host names and instead specify the IP addresses because when the host names are resolved, it isn't always possible to predict whether the results will be IPv4 or IPv6 addresses.

This type of configuration also works in reverse: if you use `interfaces` to specify IPv6 addresses only, then IPv4 will be disabled.

SMB log

Samba is basically compatible with all SMB protocol versions since 1.0. You can determine which protocols are actually supported as follows:

```
root# testparm -s -v | grep protocol
   client min protocol = SMB2_02
   client max protocol = default  (currently corresponds to SMB3_11)
   server min protocol = SMB2_02
   server max protocol = SMB3
   ..
```

This output proves that Samba requires SMB version 2.02 as a minimum standard. This default configuration of current Samba versions is part of the effort to finally eliminate SMB1 and all its security issues. Microsoft has also been advising against using SMB1 for many years. Never follow recommendations from the internet to reactivate SMB1 because of compatibility problems!

If you use an older Samba installation and `testparam` is shown as `min protocol CORE` or `LANMAN1`, you should add the following two lines to smb.conf:

```
# /etc/samba/smb.cnf
[global]
   client min protocol = SMB2_02
   server min protocol = SMB2_02
```

If your Samba version doesn't know the `SMB2_02` keyword, you can use `SMB2` instead. The client setting ensures that you do not see any network directories from old PCs or other devices on your Linux computer. This concerns old NAS devices, audio and video players, game consoles, and so on.

The server setting means that your Linux computer itself doesn't provide any directories in the old protocol. No problems are to be expected in the interaction with current Linux and Windows installations. Again, at most some older devices will no longer see your directories provided on the network.

Guests

The `map to guest = never` setting prohibits all users who cannot properly authenticate to the server from having any access. Depending on the application, it may make sense to set up directories for guests that can be read or even modified without authentication. But if that isn't necessary, you should lock out guests from the outset:

```
# /etc/samba/smb.cnf
[global]
   map to guest         = never
```

SELinux

On Fedora and RHEL, SELinux monitors all Samba activity by default. You must activate the `samba_enable_home_dirs` SELinux parameter so that home directories can be shared easily:

```
root# setsebool -P samba_enable_home_dirs on
```

If you want to share another directory, you have to set the `samba_share_t` attribute for this. To do this, run the following two commands—in the following example, for the `/samba/shares` directory:

```
root# semanage fcontext -a -t samba_share_t '/samba/shares(/.*)?'
root# restorecon -R /samba/shares
```

SELinux Configuration

There are various other SELinux parameters that control what Samba is (not) allowed to do. The `system-config-selinux` program from the `policycoreutils-gui` package provides an overview of all these parameters, including the option of changing their settings at the click of a mouse. After startup, search for "samba" in the **Boolean** dialog sheet! Further tips on SELinux can be found in Chapter 30 and, after installing the `selinux-policy-doc` package, with `man samba_selinux`.

27.2.4 Logging

Both Samba services, `smbd` and `nmbd`, log global events to the `/var/log/samba/log.smbd` and `log.nmbd` files. Neither the names nor the locations of these two logging files can be changed by `smb.conf`.

The `log file` parameter in the global section of `smb.conf` specifies where to log client-specific messages. The default `/var/log/samba/log.%m` setting causes a separate logging file named `log.hostname` to be created for each client accessing Samba. `max log size = 1000` limits the maximum size to 1,000 KiB. If a logging file becomes larger, Samba renames it to `name.old`. `syslog = 0` does not mean that syslog is not used, but that only error messages should be logged to the syslog service (`rsyslogd` or journal; in some distributions, both services are active). Alternative settings are 1, 2, 3, and so on if you also want to log warnings, notes, and debugging messages.

logrotate

You should exercise caution when using `logrotate` (see Chapter 14, Section 14.12.4). This program archives `log.smbd` and `log.nmbd` once a week in the default setting of Debian and Ubuntu and at the same time deletes all archive versions that are older than two months. `logrotate`, however, ignores the rapidly growing client-specific logging files at `log.<hostname>`.

The corresponding configuration file for Fedora and RHEL, which takes care of *all* logging files in `/var/log/samba`, is better thought out here:

```
# /etc/logrotate.d/samba on Fedora and Red Hat
/var/log/samba/*
{
    notifempty
    olddir /var/log/samba/old
    missingok
    sharedscripts
    copytruncate
}
```

Another solution is to use the following setting in `smb.conf`: `log file = /var/log/samba/log.smbd`. This allows `smbd` to log both global and client-specific messages to the same file. As long as you do not have to search for errors in the Samba configuration, this is most useful.

27.2.5 WS-Discovery (wsdd and wsdd2)

Linux and macOS computers, as well as NAS devices that share directories on the network, usually have no problems recognizing each other. Windows, on the other hand, sometimes overlooks network directories provided by Samba servers.

To improve the interaction between Linux and Windows, a WSD daemon can be installed on Linux, which makes the Samba server known on the network via the WSD protocol. Depending on the distribution, the `wsdd` Python implementation and the C variant—`wsdd2`—are available for selection. All you have to do is install the relevant package and make sure that the program gets started:

```
root# apt/dnf install wsdd[2]
root# systemctl enable --now wsdd[2]
```

27.3 Password Management

By default, `security = user` applies to Samba—in other words, user-level security. To enable the sample user `peter` to use a network directory, the following requirements must be met:

- There must be a Linux account named `peter` on the server. This account is required for access rights management. Samba thus accesses the Linux access rights to decide which user may read or modify which file.

  The Linux account doesn't need to be active. Often it's useful for security reasons to explicitly lock the account using `passwd -l peter`. This prevents Samba users from logging into the server using SSH.

- peter and the associated password must be in the Samba user database. For technical reasons, the passwords and other metadata of a Samba account are managed independently of those of the associated Linux account. There must therefore be a Linux account in parallel to the Samba account; however, the password of the Linux account is irrelevant for Samba.
- smb.conf must contain entries for network directories that peter is allowed to use (see Section 27.4).

27.3.1 Samba Passwords

Before a user can access a directory on the local network, they must identify themselves to the Samba server. When establishing a connection from a Windows client, the Windows login name and an encrypted string for the password are transmitted for this purpose.

For Linux clients, this data cannot be taken from the Linux login, which is why a separate dialog for entering the Samba user name and password usually appears. After that, the process continues as with Windows clients. Also in this case, not the password itself, but an encrypted code is transmitted to the Samba server. Only if this code matches the hash code of the password stored on the Samba server is access to the network directory granted.

smbpasswd

Thus, in order for peter to use a Samba directory, you the Linux system administrator must create the peter Samba account. The smbpasswd command with the -a (*add*) option helps you to do this. For the password, you need to enter the same string that the user peter has on Windows. To change an existing Samba password, you can use smbpasswd without the -a option:

```
root# smbpasswd -a peter
New SMB password: ******
Retype new SMB password: ******
Added user peter.

Password changed for user peter.
```

Note that sbmpasswd only works if the Linux user peter also exists on the local system (/etc/passwd file)! If necessary, you must create new Linux users upfront using useradd or adduser.

smbpasswd backend

Where is the Samba account data for peter stored? For this purpose, the Samba server has a user and password database to perform authentication. In the past, a simple text file was often used for this (passdb backend = smbpasswd).

TDB backend

Today, however, all distributions use the Trivial Database (TDB) backend (`passdb backend = tdbsam`). For really large installations with a large number of users, such as use in companies, it's advisable to manage the login data via LDAP, but I won't go any further into that in this book.

TDB is a binary format for storing data records. Its main advantage compared to the conventional `smbpasswd` files is that other data and attributes are stored along with the login name and password. These include in particular extended access control lists (ACLs), which increase compatibility with current Windows versions. For most common distributions, passwords are stored in the following file: `/var/lib/samba/private/passdb.tdb`. If necessary, you can specify a different file in `smb.conf` using `passdb backend = tdbsam:dateiname`. You can use the `smbpasswd` command to create a new Samba account or change its password. `pdbedit` provides access to any account information: `root` users can use it to create a list of all Samba users (`pdbedit -L -v`), set various attributes for each account, and more. Both commands analyze `smb.conf` and work for all password systems—that is, `smbpasswd`, `tdbsam`, and `ldapsam` (the following output is heavily shortened due to space limitations):

```
root# pdbedit -L -v

Unix username:        peter
NT username:
Account Flags:        [U          ]
User SID:             S-1-5-21-3702490679-2182029830-1044092605-1000
Primary Group SID:    S-1-5-21-3702490679-2182029830-1044092605-513
Home Directory:       \\fedora\peter
Profile Path:         \\fedora\peter\profile
Domain:               FEDORA
Password last set:    24 May 2023 11:53:10 CEST
Password can change:  24 May 2023 11:53:10 CEST
Password must change: never
Last bad password   : 0
Bad password count  : 0
...
```

27.3.2 Synchronizing Samba and Linux Passwords

In some installations, it's advisable to allow users of Samba directories to log directly into the Linux server—for example, to work via SSH and edit their files using Linux commands. In such cases, it would of course be convenient if the Samba and Linux passwords always matched. After all, it's a

real nuisance to change passwords several times (in extreme cases, three times: on the Windows client, for the Samba server, and for the Linux login).

Unfortunately, the synchronization of passwords isn't easy to accomplish because a different algorithm is used to encrypt Samba passwords than Linux passwords. For this reason, Samba and Linux passwords are managed separately. (While the algorithms are different, there is one commonality: the stored strings only allow you to check the passwords, but not to reconstruct them. It's therefore impossible to translate or convert the stored passwords from one system to another.) The most commonly used solution to this problem is to change the Linux password in parallel with each client call of `smbpasswd` to change your own password. On Ubuntu, `smb.conf` contains all the necessary settings for this:

```
# /etc/samba/smb.conf
[global]
    ...
    unix password sync = yes
    pam password change = yes
    passwd chat = *Enter\snew\s*\spassword:* %n\n
                  *Retype\snew\s*\spassword:* %n\n
                  *password\supdated\ssuccessfully* .
```

`unix password sync = yes` enables the synchronization. In this case, PAM is used because of `pam password change`. PAM modules are collections of libraries for password administration. The previously required `passwd program` parameter, which specified the path of the `passwd` program, is not relevant for PAM and is ignored. `passwd chat` describes the communication between Samba and PAM. The string has been spread across three lines in the preceding listing, but it must be specified on a single line in `smb.conf`.

Unfortunately, the synchronization has many limitations:

- `smbpasswd` must be executed by the respective user, not by `root`! Here's why: When `smbpasswd` is called by `root`, it directly manipulates the Samba user database. This works even if the Samba server isn't running at all. On the other hand, when the command is used by an ordinary user, it communicates with the Samba server, which does the actual work, including password synchronization.
- `smbpasswd` accepts any bad password, such as empty passwords or passwords consisting of only one letter. The Linux password system or PAM, on the other hand, enforces a minimum password quality and refuses overly simple passwords. This can result in the Samba password being

changed, but not the Linux password. From this point on, the passwords are no longer in sync, so any further attempt to reset the Linux password will fail. To synchronize the passwords again, `root` must reset the Linux and Samba passwords of the affected user.

- The synchronization via `unix password sync` works only in one direction: from Samba to Linux. If, in turn, a user changes his or her Linux password using `passwd`, the Samba password remains unchanged.

From my personal experience, I do not recommend using the password synchronization described here. It's error-prone and causes more problems than it solves. You should use the `unix password sync = no` setting.

27.3.3 Mapping of Windows and Linux User Names

On Windows, almost any character string of up to 128 characters is possible as a user name, whereas on Linux you can use only a character string with a maximum of 32 characters without special characters or spaces. If a Windows user uses a user name that does not directly correspond to a Linux user name, the mapping must be established via a file. The name of this file is specified by the `username map` option in `smb.conf`:

```
# /etc/samba/smb.conf
[global]
    ...
    username map = /etc/samba/smbusers
```

Each line of the `smbusers` file contains first a Linux user name, then the = sign, and finally one or more Windows user names. You need to place names with spaces in quotation marks. You can also use the file to assign a Linux user to multiple Windows users:

```
# /etc/samba/smbusers
peter  = "Peter Mayer"
...
```

An Incorrect Samba Configuration Can Put the Entire Linux System at Risk

`/etc/samba/smbusers` can break the security system of Samba or Linux if `root` is assigned to another user! Make sure that only `root` is allowed to modify the file:

```
root# chown root /etc/samba/smbusers
root# chmod 644 /etc/samba/smbusers
```

27.3.4 Putting It All Together

Let's assume that there is a Windows PC for Peter Mayer on your local network, and the login name on this machine is `Peter Mayer`. On a Linux server with Samba, there is the account named `peter`. The `smbusers` file creates the mapping between `Peter Mayer` and `peter`. Under these conditions, there are now the following combinations of login name and password:

- **Login on Windows**
  `Peter Mayer` and the Windows password.

  The Windows password is valid for local work on the Windows computer. Peter can change the Windows password on Windows.
- **Login on the server (Linux)**
  `peter` and the Linux password.

  The Linux password is valid for a direct login on the server, provided that the Linux account is active and not locked. Peter can change his Linux password—for example, after an SSH login on the server—by using the `passwd` command.
- **Access to network directories**
  `Peter Mayer` or `peter` and the Samba password.

  The Samba password applies to the use of the network directories. It should match the Windows password. If it does not, a login box appears on Windows as soon as Peter wants to access a network directory. Peter can change his Samba password after an SSH login on the server by using `smbpasswd`.

Long story short, only technically skilled users are able to change their three passwords themselves—and only if the Linux account is active. For all others the following applies: once defined, passwords should never be changed again.

From a security perspective, of course, this is anything but ideal. And this is exactly why companies usually have a central server that is responsible for all logins and their passwords (PDCs, Active Directories).

27.3.5 Working Techniques

In practice, the matter is often simpler than the preceding statements suggest. Especially on small networks, it's often sufficient to solve one of the following two tasks:

- **Sharing your own directory**
  You want to share a directory of your own, usually a subdirectory of your home directory, like `/home/<loginname>/images`. The files in this directory

belong to you, so it's obvious that you could share the directory with your account.

If the access is to be secured by a password, you would have to tell the person who is to access the images your regular Linux login password. This isn't a good idea at all!

It's usually better to share the directory for read access only, but without a password (i.e., from Samba's point of view, for any guest). Of course, this approach is not suitable for sensitive data.

- **Sharing a central directory**
  Say that you want to share a directory with files for multiple users—something password protected—but without revealing your own password. Then you need to set up a new user for this, for which no Linux login is possible. You can name this user account `shareuser`, for example.

  In the simplest case, you use `/home/shareuser` as the directory to share. Of course, you can also choose any other directory, but you must not forget to set the access rights and, if necessary, the SELinux context correctly.

  Use `smbpasswd` to assign a Samba password to `shareuser` (different from your own one, of course). Once you have populated the directory with data and set it up in `smb.conf`, you should give your employees the following information: the directory name in Windows notation (i.e., `\\hostname\directory-name`), the `shareuser` account name, and the associated password.

27.4 Network Directories

In the previous section, I explained which requirements must be met for a user to be able to log on to Samba; in short, a Linux account and a Samba password must be available. Now the only question is which resources a logged-in user can see and use. The `[<resourcename>]` sections in `smb.conf` are crucial for this.

User directories

The definition of a directory that can be accessed by a specific user looks as follows:

```
# in /etc/samba/smb.conf
...
[directory1]
   valid users = peter
   path        = /data/dir1
   writeable   = yes
```

With this setting, the user `peter` and all users assigned to this Linux account by `smbusers` can read and modify the `/data/dir1` directory. In the file manager, this resource is named `directory1`—that is, the string specified in square brackets.

The meaning of the keywords is easy to understand: `valid users` specifies the username. You're allowed to specify multiple comma-separated usernames. `path` specifies which directory of the server will be shared. If `path` is not explicitly specified, then Samba defaults to the home directory of the specified user. `writeable = yes` allows changes within the directory. Without this option, the user has read-only access.

Basically, Linux access rights apply to all accesses. So if there is a file in `/data/dir1` that belongs to `root`, then the Linux user `peter` is normally only allowed to read this file, but not to modify it. This restriction also applies to all users of the network directory.

SELinux Can Prevent Access to Directories

If you set up your Samba server on Fedora or RHEL, you should also think about the already mentioned SELinux security system! You must assign the `samba_share_t` attribute to the directory to be shared to make sure the access works:

```
root# semanage fcontext -a -t samba_share_t "/data/dir1(/.*)?"
root# restorecon -R /data/dir1
```

Home directories

When setting up a file server for a large number of users, the easiest thing to do is to allow each logged-in Samba user to directly view and edit their Linux home directory. Instead of replacing `smb.conf` with countless entries like the following:

```
[username]
   path        = /home/username
   valid users = username
   writeable   = yes
```

`smb.conf` instead provides the following short notation, which is also included by default in `smb.conf` on some distributions (e.g., Fedora):

```
[homes]
    valid users  = %S, %D%w%S
    browseable   = No
    read only    = No
```

This will make the home directory of the currently active user visible under their name. The `browseable = no` option does not, as one might think, cause

the user not to see his directory. It only prevents the directory from being visible twice: once under the respective user name (such as `peter`) and once as `homes`.

When `valid users` is set, `%S` refers to the logged-in user. `%D%w%S` is an alternate notation that includes the domain (`%D`) and the host name or IP address (`%w%S`).

Once Again, SELinux!

The default configuration of SELinux on Fedora and RHEL allows sharing of home directories only if the Boolean parameter `samba_enable_home_dirs` is set:

```
root# setsebool -P samba_enable_home_dirs on
```

Group directories

User and home directories allow users to store their files centrally on the server, but do not provide a way to share data. The directories are invisible and inaccessible to other users. This can be solved by group directories that all members of a group are allowed to use. The group assignment is carried out by the Linux user management. The group is specified with the `user` keyword in the `@groupname` notation:

```
# in /etc/samba/smb.conf
...
[salesdata]
   valid users    = @sales
   path           = /data/sales
   writeable      = yes
   force group    = +sales
   create mask    = 0660
   directory mask = 0770
```

For access to group directories, the correct setting of access rights for files and directories is particularly important. This also applies to files and directories that are newly created. `force group = +sales` causes newly created files or directories to be assigned to the `sales` group and not to the user's default group as is usually the case. If a user is not a member of the `sales` group, they are not allowed to access the directory.

Beware of "force group"

In a group assignment as in the previous example, you must never use the `force group = sales` setting, without a preceding plus sign! This would result in Samba performing every access to the directory as if the currently

active user were a member of the `sales` group—even if the Linux-level user is not a member of this group at all!

In other words, you can use `force group = sales` to give users who do not belong to the `sales` group read and write permissions for the directory. With group directories, this is rarely intentional and can be a major security issue!

The `create mask` and `directory mask` parameters ensure that files and directories newly created by group members can be read and modified by all other group members. The octal number corresponds to the `chmod` value (see `man chmod`). If new files or directories may only be read by other group members, but not modified, you should use the values 0440 and 0550.

Freely accessible directories

Access to the `share` directory is even more permissive: any user who can authenticate with Samba can read files from this directory. Write access is disabled in the following example:

```
# in /etc/samba/smb.conf
...
[share]
   path      = /data/share
   read only = yes
```

Of course, you can also set up freely accessible directories with write access (`writable = yes`). By default, all users can read but not modify the files created by other users. The solution is to set `force group` and the two `mask` parameters. Unrestricted mutual read and write permissions are achieved by using `create mask = 0666` and `directory mask = 0777`.

Access for unauthenticated users

All of the preceding examples assume that the user can authenticate with Samba. If configured appropriately, Samba also provides directory access for unauthenticated users. Such users are called *guests* (`guest` users) in Samba terminology. The global settings summarized in the following listing are responsible for managing guests:

```
# in /etc/samba/smb.conf
[global]
   ...
   map to guest           = bad user
   guest account          = nobody
```

`map to guest = bad user` causes login attempts with a nonexistent username to be automatically mapped to the virtual `guest` Samba user. By default, however, there are no network directories or other resources that `guest` is allowed to use. `guest account` specifies to which Linux user guest is

assigned. In most Linux distributions, including Debian and Ubuntu, the user `nobody` is provided for this purpose.

You should mark directories that should be usable by guests with `guest ok = yes`. As a rule, you will protect such directories from write access using the `read only = yes` attribute. Remember that guests are generally only allowed to read or modify files that the Linux user `nobody` is also allowed to read or modify:

```
[guest]
   path        = /data/guest
   guest ok    = yes
   read only   = yes
```

A variant of `guest ok` is `guest only = yes`. With this setting, the directory can be used only by guests, not by authenticated users. If you do not want to grant guests access to Samba resources in general, you can use the `map to guest = never` setting in the `[global]` section.

User shares

Samba offers ordinary users without `root` privileges the possibility to share directories themselves, as so-called user shares. The corresponding configuration files are usually stored in the `/var/lib/samba/usershares` directory, with each shared directory getting its own file. The following lines represent an example of this:

```
# file /var/lib/samba/usershares/images
path      = /home/kofler/images
comment   =
guest_ok  = yes
sharename = Images
```

Details of the user share configuration are controlled by various `usershare` statements in the global section of `smb.conf`. `usershare allow guests` allows user shares to be shared for use by the `guest` account, without password protection. This requires `guest ok` or `guest only` to be specified when you define the directory. `usershare max shares` limits the number of user shares. The lines of the following listing reflect the default settings of Samba:

```
# in /etc/samba/smb.conf
[global]
   ...
   usershare allow guests = yes
   usershare max shares   = 100
```

Recycle bin for Samba directories

When files in network directories are deleted, they are usually lost forever. The recycle bin functions provided by Linux, Windows or macOS, each

apply only to local files. However, to avoid the unwanted loss of files, you can also set up a recycle bin at the Samba level. In the simplest configuration, a single line in `smb.conf` is sufficient for this:

```
[global]
  vfs objects = recycle
```

This enables the Samba VFS extension module `recycle`, where VFS stands for *virtual file system*. Deleted files now end up in the `.recycle` directory by default (relative to the share directory). However, this directory is invisible in most file managers. To make the process more transparent, you can use `recycle:repository` to specify a different directory name for the recycle bin—for example, as follows:

```
  recycle:repository = recycle bin
```

Now files will only be actually deleted when they are also deleted from the recycle bin. Under certain circumstances, it may be useful to run a script on the Samba server that automatically deletes old files from the recycle bin after a certain period of time. For this to work, however, you must update the modification date when moving the files to the recycle bin (`recycle:touch = Yes` option). You can find various other `recycle:xxx` options in `man vfs_recycle`.

27.4.1 Sharing Network Directories in GNOME and KDE

When desktop users want to quickly and easily share a directory via Samba, they usually don't want to make manual changes to `smb.conf`. That is why the file managers of GNOME and KDE provide dialogs for sharing directories.

Behind the scenes, Nautilus (GNOME) and Dolphin (KDE) use Samba's usershare mechanism. The parameters for each shared directory are stored in a separate configuration file in the `/var/lib/samba/usershares` directory. The access rights of this directory are set in such a way that all users belonging to a certain group (`sambashare` on Ubuntu) are allowed to create new files in it.

Know this much in advance: don't get your hopes up too high that directory sharing will be straightforward. If available at all, the GNOME and KDE tools require basic configuration of the Samba server, and depending on the distribution, you must not forget the firewall and SELinux configuration.

But even then, the sharing tools remain piecemeal because they don't care about Samba passwords. The tools only work satisfactorily if both Samba in

general and the sharing of directories by the respective user have already been set up anyway: under these conditions, it would then be a piece of cake to share another directory on the network.

Preparations

Ahead, I assume that the Samba server has been installed—usually the `samba` package. If a firewall is active, you must open the Samba ports there. The user who is to share network directories must belong to the `sambashare` group. This setting will take effect only with the next login:

```
user$ sudo usermod -a -G sambashare <username>
```

If you use SUSE or openSUSE, you must perform a basic Samba configuration before sharing the first directory on KDE or GNOME.

You can use the **Network Services • Samba Server** YaST module for this purpose. In the **Identity** dialog sheet, you should keep the **No DC** default setting as the domain controller. The other variants are only relevant if Samba is also supposed to manage login data for Windows accounts.

Then select the **Start on Boot** option in the **Start** dialog sheet to start the Samba server automatically in the future. To allow other computers on the local network to access the network directories, you must also enable the **Open Port in Firewall** option.

Nautilus/GNOME

Provided that the `nautilus-share` package is installed, you can share directories in the GNOME file manager. To do this, right-click the directory and run the **Share Options** command from the context menu. This will take you to the dialog shown in Figure 27.1.

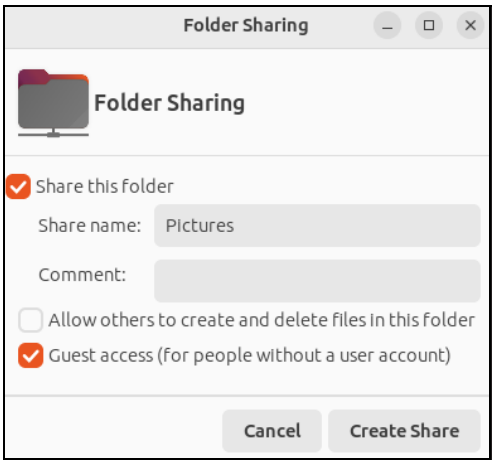


Figure 27.1 Sharing Directory on GNOME (Debian/Ubuntu)

Unfortunately, Fedora and RHEL users have to do without `nautilus-share`, probably because this Nautilus extension is not compatible with the Red

Hat-specific SELinux configuration. In the past, the `system-config-samba` tool from the package of the same name was available as an alternative. However, this program is no longer maintained and is therefore also missing from the package sources. So there is no way around editing the configuration files manually.

Dolphin/KDE

On KDE, you can set up a network directory in the **Share** tab of the properties dialog in the Dolphin and Konqueror file managers. For this to work, the `kdenetwork-filesharing` package must be installed.

In the terminal

If there are no graphical tools, you can also create user shares in the terminal by using the `net usershare` command. Again, the same requirements apply; that is, Samba must have been installed and you must be a member of the `sambashare` group:

```
user$ net usershare add myshare /home/kofler/Documents/
```

Samba Passwords Must Be Defined Manually!

Finally, there is the question of who is allowed to use the shared directories. If you have enabled the **Guest Access** option, everyone will have access to your network directory without logging in.

On the other hand, if you do not enable the option, the user will have to log in with a user name and password. However, this only works for users for whom there is an account on the relevant computer, and only if a Samba password has been defined via `smbpasswd` for those users. Neither KDE nor GNOME cares about this. If necessary, you must set your Samba password using `smbpasswd` in a terminal window.

Shares in the GNOME Settings

In the system settings of GNOME you can find the **Shares** module. This module allows you to share the `Images`, `Music`, `Public`, and `Videos` directories within your home directory on some distributions. However, these shares have nothing to do with Samba!

Rather, GNOME launches a local instance of the Apache web server for your public directory, which provides the respective directory using the WebDAV protocol over the desired network interface. For the remaining directories, an instance of a DLNA server will be started. The files can only be accessed by suitable programs, such as Linux file managers (WebDAV) or media players (DLNA).

27.5 Example: Home and Media Server

This section provides a simple example of how you can configure a central Samba server. The starting point is a computer-savvy household in which more and more data is accumulating on three computers belonging to parents and their two children: photos, audio and video files, school documents, accounting, and so on. The decentralized storage of data poses some problems:

- There are no proper backups. What happens if a notebook is lost or is damaged on the way to school?
- It's difficult to access shared data. Grandma should get an album of the best family photos from previous years for her next birthday. However, the digital photos are spread across three computers and are not organized in any way.
- The exchange of data between computers is cumbersome and is usually done with the help of a USB flash drive.

The Linux-enthusiastic son finally suggests solving these problems with a central home or media server. The server can be integrated into the home network via Wi-Fi and can also take over other functions if required.

At this point, only the Samba configuration is of interest. Each family member gets their own network directory where they alone are allowed to write and read data. The data stored there is therefore private, although of course it must be clear to all family members that the son, as the administrator, can ultimately read and modify any file. There are also five other directories for a common data exchange. Parents can access `parents`, children can access `children`, and all family members can access the `family`, `audio`, and `photos` directories. Of course, it would have been possible to simply set up the `audio` and `photos` directories as subdirectories of `family`, but defining your own network directories makes the application a bit more intuitive. If required, of course, any other users and directories can be set up.

Setting up Linux users and groups

In the following, I will use `mother`, `father`, `daughter`, and `son` as usernames. In real life, of course, you will use real names here—but I have refrained from doing so here, so that you do not have to learn the names of a fictitious family as well:

```
root# groupadd parents
root# groupadd children
root# groupadd family
root# useradd --create-home --groups parents,family father
```

```
root# useradd --create-home --groups parents,family mother
root# useradd --create-home --groups children,family son
root# useradd --create-home --groups children,family daughter
```

Because useradd was executed without a password, the new users are automatically locked; that is, they cannot log on. That is intended. It is neither necessary nor convenient for family members to log on to the server directly.

Together with each user, a new group with the same name is automatically created, which is the default group for the user. In addition, the new users are also assigned to the family and parents or children groups. father thus belongs to the father, parents, and family groups; mother to the mother, parents, and family groups; and so on. If you want to assign another group to a user later, the easiest way is to use the following command:

```
root# usermod -a -G newgroup user
```

Setting up Samba users

Next, the Samba users need to be set up, this time each with a password:

```
root# smbpasswd -a father
root# smbpasswd -a mother
...
```

Setting up directories

When you set up the directories for the shared files, it's important to set owners and access rights correctly; otherwise the data access will not work later. The chmod 770 command ensures that only group members are allowed to read and modify the directory. The second command prohibits access to the home directories by other users:

```
root# mkdir /shared-data
root# mkdir /shared-data/{parents,children,family,audio,photos}
root# cd /shared-data
root# chown :parents parents/
root# chown :children children/
root# chown :family family/ audio/ photos/
root# chmod 770 *
root# chmod 770 /home/{father,mother,son,daughter}
```

One advantage of the shared file server is the ability to create centralized backups. You only need to back up the /home and shared-data directories.

Samba configuration

The following lines present the smb.conf configuration file. Password synchronization and any Samba access by guests are disabled. The settings for the various directories should be understandable without further explanation after reading Section 27.4:

```
# /etc/samba/smb.conf for a home server
[global]
   workgroup          = home
   server string      = %h server (Samba, Ubuntu)
   security           = user
   passdb backend     = tdbsam
   unix password sync = no
   invalid users      = root

   map to guest       = never
   log file           = /var/log/samba/log.%m
   max log size       = 1000
   syslog             = 0
   dns proxy          = no
   panic action       = /usr/share/samba/panic-action %d
[homes]
   browseable         = no
   writeable          = yes
[parents]
   valid users        = @parents
   path               = /shared-data/parents
   writeable          = yes
   force group        = +parents
   create mask        = 0660
   directory mask     = 0770
[children]
   valid users        = @children
   path               = /shared-data/children
   writeable          = yes
   force group        = +children
   create mask        = 0660
   directory mask     = 0770
[family]
   valid users        = @family
   path               = /shared-data/family
   writeable          = yes
   force group        = +family
   create mask        = 0660
   directory mask     = 0770
[photos]
   valid users        = @family
   path               = /shared-data/photos
   ... as with [family]
```

```
[audio]
   valid users        = @family
   path               = /shared-data/audio
   ... as with [family]
```

The configuration has a small flaw: all users see *all* shares, even those which are not intended for them and which they are not allowed to use. For example, the children see the `parents` directory, and the parents see the `children` directory. Actual use of these directories fails as planned due to access rights, but of course it would be even more elegant if those directories were not visible in the first place.

Unfortunately, Samba doesn't offer any configuration options to that end. You can hide individual directories by `browseable = no`, but then no one will see the directories anymore, not even the legitimate users. The directories remain usable, but the correct path must be specified manually.

The `hide unreadable = yes` and `hide unwriteable = yes` options do not help either. They hide all files *within* a network directory that a user cannot read or modify. However, the network directory itself remains visible.

27.6 Example: Company Server

In this example, a small company manufactures measuring devices. A Samba server is set up to simplify central backups and data exchange. All employees get their own directory there. There are also several shared directories that are summarized in Table 27.2. Table 27.3 shows to which groups the employees belong depending on their position in the company.

Network Directory	Access (Groups)
`strategy`	`management`
`accounting`	`accounting, management`
`development`	`development, management`
`sales`	`sales, accounting, management`

Table 27.2 Data Exchange on the Company Server

Employee	Group Membership
Management	`management, development, sales, accounting`

Table 27.3 Group Membership of Employees According to Their Position

Employee	Group Membership
Accounting/controlling	accounting, sales
Developers	development
Sales team, marketing	sales

Table 27.3 Group Membership of Employees According to Their Position (Cont.)

Less Is More!

If you need to create a group and directory schema yourself, simplicity should be your top priority. The more groups and directories there are, the more confusing the system will become, the more unwieldy its application, and the more tedious its maintenance.

Even if you implement the Samba server with high-quality hardware and use gigabit network connections, access to network directories will be slower than to local files. That is why some users will continue to store and edit large files locally for performance reasons. In this case, there should definitely be an automatic script that synchronizes all local files with the user's private network directory once a day. All the advantages of a central backup system are lost when individual users save their files on the local PC or notebook computer!

Setting up Linux users and groups

For the sake of simplicity, I will use `bossN`, `developer`, and so on as user names in the following sections. In real life, of course, you will use real names. Except for the names, the commands for setting up the groups and users are similar to those in the previous example (home server). A direct login of the employees on the server is not intended, which is why the password specification is omitted at this point:

```
root# groupadd management
root# groupadd sales
root# groupadd accounting
root# groupadd development

root# useradd --create-home \
        --groups management,sales,accounting,development boss1
root# useradd --create-home --groups sales sales1
root# useradd --create-home --groups sales sales2
root# useradd --create-home --groups development developer1
root# useradd --create-home --groups development developer2
```

```
root# useradd --create-home --groups accounting,sales accountant1
root# useradd --create-home --groups accounting,sales controller1
root# ...
```

Setting up Samba users

Next, the Samba users need to be set up, this time each with a password:

```
root# smbpasswd -a boss1
root# smbpasswd -a sales1
...
```

Setting up directories

When setting up the directories for the shared files, it's important to set owners and access rights correctly; otherwise the data access won't work later. The `chmod 770` command ensures that only group members can read and change the directory and prevents other users from accessing the home directories:

```
root# mkdir /companydata
root# mkdir /companydata/{development,sales,accounting,strategy}
root# cd /companydata
root# chown :development development/
root# chown :sales    sales/
root# chown :accounting accounting/
root# chown :management     strategy/
root# chmod 770 /companydata/* /home/*
```

Samba configuration

With the exception of the `workgroup` string, the global section of `smb.conf` looks exactly the same as in the home server example (see Section 27.5). The definition of the network directories is also very similar to the previous example. The difference is that members of multiple groups are allowed to access the directories. To ensure that the data exchange functions smoothly, `force group` is particularly important. All users who are allowed to access a directory must be members of this group. Don't forget the plus sign in front of the group name in `smb.conf`:

```
# /etc/samba/smb.conf for a company server (standalone configuration)
[global]
   ... as in the previous example (home server)
[homes]
   browseable     = no
   writeable      = yes

[strategy]
   valid users    = @management
   path           = /companydata/strategy
   writeable      = yes
```

```
   force group    = +management
   create mask    = 0660
   directory mask = 0770

[accounting]
   valid users    = @accounting, @management
   path           = /companydata/accounting
   writeable      = yes
   force group    = +accounting
   create mask    = 0660
   directory mask = 0770

[development]
   valid users    = @development, @management
   path           = /companydata/development
   writeable      = yes
   force group    = +development
   create mask    = 0660
   directory mask = 0770

[sales]
   valid users    = @sales, @accounting, @management
   path           = /companydata/sales
   writeable      = yes
   force group    = +sales
   create mask    = 0660
   directory mask = 0770
```

27.7 SMB Client Access

This section deals with the question of how a client PC accesses the network directories provided by Samba, by a Windows computer, or by a NAS device.

Firewall

Before you get annoyed with Windows, Linux, SMB, or anything else, it's essential to make sure that there is not a firewall blocking access to the network directories. The TCP ports 135, 139, and 445 and the UDP ports 137 and 138 are decisive.

If you run Debian or Ubuntu, there is not much risk in this regard; by default, these distributions do not run a firewall. Current Fedora versions do run a firewall, but the ports relevant for client access are excluded. You only need to change the firewall configuration if you want to share directories yourself (i.e., when using the server).

On SUSE and RHEL, on the other hand, a firewall is active by default, which blocks the ports relevant for SMB—for both client and server access. The following commands provide a solution to this problem by first determining the active firewall zone and then setting an exception rule for it:

```
root# firewall-cmd --get-active-zones
public
  interfaces: enp1s0
root# firewall-cmd --zone=public --permanent \
  --add-service=samba-client
root# firewall-cmd --reload
```

Avahi

For Linux to "see" the other machines on the local network, your Linux machine should be running `avahi` (see Chapter 15, Section 15.5). In most desktop distributions, this is automatically the case. If a check using `systemctl status avahi-daemon` does not provide a positive result, you should install the `avahi` package and check that the service is running by using `systemctl enable --now avahi-daemon`.

27.7.1 Using Desktop Systems

Before you can access Windows or Samba network directories on Linux, you need to make sure that the Samba client tools are installed. This is often the case by default. On Ubuntu, the required programs are packaged in `samba-common`, `smbclient`, and `libsmbclient`; on Fedora, in `samba-common`, `samba-client`, and `samba-libs`.

The ideal case: directory access by mouse click

With a little luck, the file managers of GNOME (**Other Places**) or KDE (**Network • Shared Folders**) will show all devices/servers on the local network that offer directories. A few mouse clicks and, if necessary, the entry of the login data (user name and password) then lead to the desired directory.

Last time, I tested this on Ubuntu 23.04 and Fedora 38—and basically, almost everything worked fine (see Figure 27.2). But do not click the promising **Windows Network** item! This item only works for SMB1 directories, so nowadays it almost never does.

Unfortunately, real Windows computers are often missing from the list of network devices. However, if you know their host names, access is not difficult either: you can specify the server address as `smb://servername` or `smb://servername/directoryname` in the **Connect to Server** line or by pressing Ctrl+L. If you don't make a mistake, the login dialog for directory access will appear shortly after.

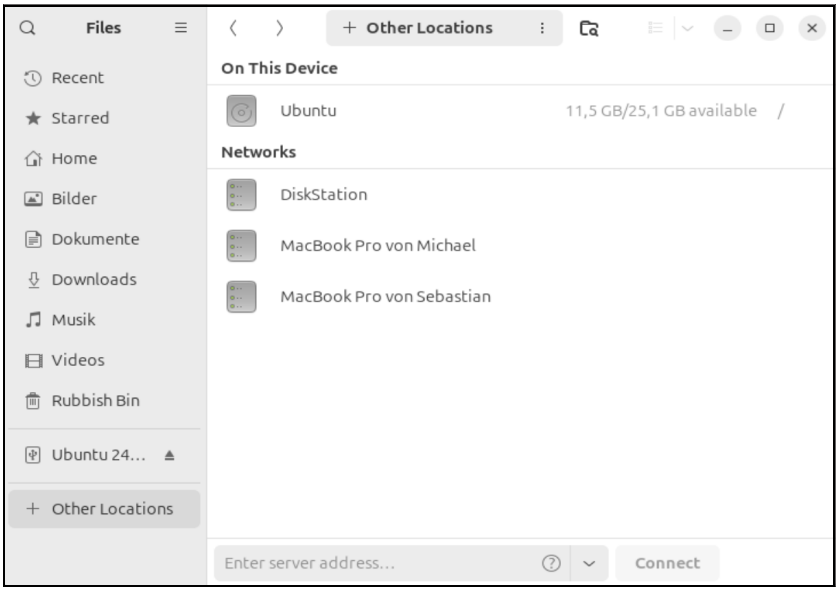


Figure 27.2 GNOME File Manager: List of Linux Servers, Apple Computers, and NAS Devices Accessible on Local Network

Last resort: terminal

A simple Linux rule is: "There is *always* a solution in the terminal." If problems occur at the desktop level, I will introduce you to some commands on the following pages that can help: `nmblookup` and `smbclient` help with troubleshooting, and `mount -t cifs` succeeds in integrating a network share into the local directory tree even in stubborn cases.

27.7.2 Finding a Samba Server Using nmblookup

Which devices or servers provide SMB shares on the local network? In the past, `smbtree` used to answer this question. However, this useful command requires the browsing features of SMB1 and is therefore obsolete on modern networks. The little-known `nmblookup` command from the `samba-common-bin` package (Debian/Ubuntu) or `samba-client` (Fedora/RHEL) steps into the breach here. It returns the IPv4 addresses to the specified NetBIOS name. Instead of the name of a computer, you can also pass `'*'` or `WORKGROUP` as parameters:

```
user$ nmblookup WORKGROUP
192.168.178.50 WORKGROUP<00>
192.168.178.28 WORKGROUP<00>
...
```

With the additional `-S` option, the command reveals the associated NetBIOS names for each device:

```
user$ nmblookup -S WORKGROUP
192.168.178.50 WORKGROUP<00>
    U2104           <00> -         B <ACTIVE>
    WORKGROUP       <1e> - <GROUP> B <ACTIVE>
        ...

192.168.178.28 *<00>
    DISKSTATION     <00> -         B <ACTIVE>
        WORKGROUP       <00> - <GROUP> B <ACTIVE>
        ...
```

A clearer result is provided by the command with the `-T` option. In my tests, however, the resulting list was then incomplete:

```
user$ nmblookup -T 'WORKGROUP'
u2104.fritz.box,       192.168.178.50 WORKGROUP<00>
DiskStation.fritz.box, 192.168.178.28 WORKGROUP<00>
raspberrypi.fritz.box, 192.168.178.40 WORKGROUP<00>
...
```

27.7.3 Access to Network Directories Using smbclient

smbclient

You can also use `smbclient` to browse network directories. The command offers little comfort but is often useful for tracking down Samba issues. Once you have used `nmblookup` to determine the names of the NMB servers on the local network, `smbclient -L <hostname>` displays all the shared resources of the specified host:

```
user$ smbclient -L win10
Enter SAMBA/user's password: ******
   Sharename       Type      Comment
   ---------       ----      -------
   ADMIN$          Disk      Remote Administration
   C$              Disk      Default Share
   IPC$            IPC       Remote IPC
   Users           Disk
```

If `smbclient` returns a login error message (*session setup failed: NT_STATUS_ACCESS_DENIED*), your current Linux user name does not match the user names of the Windows computer or Samba server. The simplest solution is to pass the user name (`-U` option) and, if necessary, the workgroup name `-W`) to the command:

```
user$ smbclient -U <username> -W <workgroupname> -L <hostname>
Enter SAMBA/username's password: ******
```

If you're interested in resources that are accessible without login, you should pass an invalid user name (such as `-U none`) and suppress the password request by using the `-N` option. Note that depending on the configuration of the server, network directories may also be listed in this case, access to which will later require a login after all.

Only in rare cases is it necessary to limit the highest possible protocol version with the `-m` option. Thus, `-m SMB2` means that you do not want to use `SMB3`:

```
user$ smbclient  -U none -N -m SMB2 -L libreelec
```

You can also use `smbclient` interactively to transfer files. To do this, first connect to the Windows computer or Samba server for the shared directory. You must specify the directory in the Windows-typical notation: `\\servername\directoryname`. To prevent the `\` characters from being processed by the shell, they must be duplicated. It's easier to use the `/` characters, which are more common on Linux and which `smbclient` fortunately also accepts.

Then, as with the `ftp` command, you can view directories via `ls`, change directories using `cd`, transfer files to the local computer by using `get`, and save files to the external computer via `put`. You can get an overview of the most important commands via `help`. A detailed description of the commands can be found in `man smbclient`:

```
user$ smbclient -U kofler //diskstation/archive
Enter WORKGROUP/kofler's password: ******
Try "help" to get a list of possible commands.
smb:/> ls
  .            DA        0  Mon Mar 22 14:51:28 2021
  ..           DA        0  Fri Apr 16 07:01:18 2021
  audio        DA        0  Sun Mar 13 21:36:44 2016
  bak          DA        0  Sun Dec  3 18:16:27 2017
  code         DA        0  Sat Aug  1 16:43:18 2020
  ...
  1290153920 blocks of size 1024. 396335724 blocks available
smb:/> get time-recording.ods
getting file /time-recording.ods of size 64487 ...
smb:/> exit
```

27.7.4 CIFS-mount

`smbclient` is a tool to locate SMB resources in case of connection issues and to transfer individual files if necessary without establishing a permanent

connection to the network directory. To be able to work persistently and comfortably, you have to integrate the network directory into the local directory tree by using `mount`.

CIFS and SMBFS

The underlying technology is called *Common Internet File System* (CIFS). This is the successor to *SMB File System* (SMBFS). CIFS-compatible Samba servers pass on Unix-/Linux-compatible information about file access rights, which was not possible with SMBFS.

mount

CIFS requires that the Samba client tools and CIFS support for `mount` and, in particular, the `mount.cifs` command be available. For Debian and Ubuntu, the `cifs-utils` package must be installed for this purpose. Fedora and openSUSE have the package installed by default.

To mount an external directory, you need to specify one of the following two commands, depending on whether Windows sharing is based on user names or not:

```
root# mkdir /media/archive
root# mount -t cifs //diskstation/archive /media/archive
root# mount -t cifs -o username=<name> //diskstation/archive \
      /media/archive
```

This mounts the directory named `archive`, which is shared on the `diskstation` NAS device into the Linux file system.

The data is now available to the `root` user under the `/media/archive` Linux directory. When you run the command, you must enter parameters. You can also specify the password directly, but this is insecure:

```
root# mount -t cifs -o username=name,password=xxxxxxxx \
      //diskstation/archive /media/archive
```

More mount options

In real life, many more options are often required before the network directory can be used correctly. For example, a typical `mount` command might look as follows:

```
root# mount -t cifs \
      -o username=name,uid=1000,gid=1000,vers=2.1 \
      //diskstation/archive /media/archive
```

For a summary of the main `mount` options, see Table 27.4.

For a better understanding of the previous command, a few more explanations are necessary: To enable you to read and write to the network directory not only as `root` but also as an ordinary user, you must specify your personal user and group identification numbers in the `mount` command

using uid=n and gid=n; you can quickly determine these IDs via the id command. In addition, dir_mode and file_mode specify the access rights with which the network directories and files should be visible. Access rights are specified in an octal notation, which is explained in more detail in Chapter 9, Section 9.7. The most useful settings are 0700 and 0600. This allows the user who is named via uid to edit all directories and read and modify all files. Other users of the computer, however, do not have access.

The nounix option seems a little absurd: the CIFS standard has been extended by the so-called Unix extensions in order to achieve better compatibility with Unix/Linux systems. On some NAS devices, these extensions are also active. In this case, mount takes the user and group information and access rights from the NAS device and ignores the uid, gid, dir_mode, and file_mode options.

However, this is only useful if this data is coordinated between the NAS device and the clients. Because this is rarely the case, the Unix compatibility mode must be disabled so that uid, gid, dir_mode, and file_mode remain effective. Sometimes you can also avoid incompatibilities with nounix, which express themselves via a vague error message: *operation not supported*.

iocharset=utf8 avoids the incorrect display of file names with international characters.

Option	Meaning
credentials=file	Reads the login name and password from a file
dir_mod=n	Sets the access bits for directories
domain=name	Determines the workgroup or domain
file_mod=n	Sets the access bits for files
gid=n	Specifies which group gets access to the files
iocharset=utf8	Uses the UTF8 character set for file names
nounix	Disables the Unix extensions of the CIFS protocol
password=xxx	Specifies the password (not secure!)
uid=n	Specifies who gets access to the files
username=name	Specifies the username
sec=ntlm	Sets the authentication mode
vers=n	Sets the version number of the SMB protocol

Table 27.4 CIFS mount Options

The sec=ntlm option is necessary for interaction with old NAS devices or Samba servers. By default, mount uses the ntlmssp method for authentication. Older servers do not support this method, in which case you have to accept the less secure ntlm method.

A similar function is fulfilled by vers=n: If you want to connect to old NAS devices, this may fail because mount only uses the SMB protocol in a current version. If you want to explicitly use an older version, you need to specify it using vers—for example, as vers=2.1.

You can obtain even more details on the permitted options via man mount.cifs.

Simplifying access using /etc/fstab

Usually, two or three attempts are required to find an ideal combination of mount options. In the future, however, you will certainly not want to type a two-line mount command every time. If you work on a desktop computer that has permanent access to an always-on server or NAS device, you can automatically include the network directory in the directory tree. To do this, you want to add an appropriate line to /etc/fstab, like the following one, for example:

```
# in /etc/fstab
//diskstation/archive /media/archive cifs \
nofail,username=u,password=p,...  0 0
```

All CIFS directories named in /etc/fstab are included during the init process. The nofail option prevents the start process from hanging up due to an unavailable network directory.

Swapping out login data

For security reasons, specifying the password directly in /etc/fstab is not recommended. A little better is to outsource this information to a separate file that can only be read by root. Therefore, you should set up the /etc/.winshare-pw file, which contains the login name, password, and optionally the workgroup (domain) for the network directory. The structure of the file looks as follows:

```
username=name
password=xxxx
domain=workgroup
```

The following command restricts access to the file so that only root can read and modify it:

```
root# chmod 600 /etc/.winshare-pw
```

Then add the `credentials` option or its short variant, `cred`, to the `fstab` entry. When mounting the directory, the authentication files are read from `.winshare-pw`:

```
# Addition to /etc/fstab
//diskstation/archive /media/archive cifs cred=/etc/.winshare-pw,... 0 0
```

Including a directory only when needed

If your Linux machine is a notebook or if the network directory is only sometimes available (e.g., because the NAS device is not running continuously but is only turned on or woken from sleep mode when needed), you should opt for a semimanual solution. You can use the following `fstab` line as a guide:

```
# /etc/fstab
//diskstation/archive /media/diskstation cifs noauto,users,cred=... 0 0
```

Two new options are crucial for this functionality: `noauto` prevents the network directory from being activated immediately during the boot process, and `users` also allows the `mount` process for ordinary users without `root` privileges. You can then run `mount` and `umount` as needed and only need to specify the mount directory:

```
user$ mount  /media/diskstation
user$ umount /media/diskstation
```

However, the `fstab` entry is also analyzed by the GNOME file manager, which displays the **Archive** entry for the network directory of the same name in the sidebar. One mouse click is enough to activate the network directory; you can deactivate it again with a second mouse click. Even old GNOME versions work with this configuration, as you would expect from the beginning!

Timeout problems

Network directories mounted using `mount` cause problems if the SMB device is suddenly no longer accessible—for example, because the affected computer has been switched off. Then, no matter whether you work in the file manager or in the terminal, longer delays will occur during which the program or command does not respond. Even when shutting down the computer, you have to wait several minutes for Linux to realize that any pending operations in the open network directory cannot be completed. The solution is to run `sudo umount -f <mount directory>`, where `-f` stands for *force*. And even this command first takes a longer pause before it's finally completed.

PART VII

Security

Chapter 28
Backups

Almost 20 years ago, Apple proved with its Time Machine that even a backup program can generate enthusiasm. Linux unfortunately cannot keep up in this respect: the range of backup commands and tools is large, but desktop users are still waiting for the ingenious, easy-to-use backup tool.

I'm not even going to try to convince you of the need for backups here. Instead, I want to focus on introducing you to various ready-to-use programs as well as tools and techniques for programming your own backup solutions:

- Backup user interfaces: Déjà Dup, Back In Time, and Grsync
- Backup command: Borg Backup
- Developing custom backup scripts: `gzip`, `tar`, `rsync`, `rdiff-backup`, `rsnapshot`, LVM, and S3

Of course, all programs or commands are open-source software. It is up to you which tools you use in which combination; there is no universal recipe for this. The ideal backup strategy depends too much on the nature of the data, the type of computer (desktop PC/notebook/server), the backup medium (external hard disk, network directory, cloud), the type of implementation (user interface, Cron script), the internal processes (compression, encryption, deduplication), and many other factors.

Further Reading

A good overview of backup programs for Linux, including a discussion of the techniques used, can be found in the blog article at *https://www.lullabot.com/articles/backup-strategies-for-2018*. While the author focuses a lot on backups in the cloud, much of the information summarized in the article is still generally applicable despite the fact that it was published a long time ago.

28.1 Déjà Dup

Déjà Dup is probably the most popular backup tool for Linux with a graphical user interface. It is set up as the default backup program by many distributions. Déjà Dup is designed to back up the home directory as easily as

possible to a local directory or a directory that can be accessed via SSH. The program is not suitable for backing up system files.

Déjà Dup performs a full backup when run for the first time and incremental backups thereafter. The backed-up files are compressed and, if desired, encrypted and then saved in a file format of your choice. They can only be extracted by Déjà Dup itself (or by using the underlying `duplicity` command).

Benefits and drawbacks

The greatest advantages of Déjà Dup include its extremely simple operation and good integration into the GNOME desktop. I don't know of any other backup program that is so easy to set up on Linux and that gives such easy access to the backed-up files.

The disadvantage, however, is the very complex internal backup technique based on `duplicity`. Performing regular backups as well as restoring individual files or directories takes an above-average amount of time and involves a considerable CPU load. If the backup is stored on the network or in the cloud, large amounts of data must be downloaded even to restore individual files.

You can find further information at the following pages and at the following Reddit post, which very succinctly summarizes the disadvantages of the `duplicity` format:

- *https://wiki.gnome.org/Apps/DejaDup*
- *https://duplicity.gitlab.io*
- *https://www.reddit.com/r/linux/comments/ahpdzk/borgbackup_frontends/eehtrfd*

28.1.1 Configuration and Use

In many distributions with GNOME as the desktop system, you start Déjà Dup by selecting the **Data Backup** program name. You may need to install the `deja-dup` and `deja-dup-nautilus` packages prior to that.

Determine where to back up

Déjà Dup is configured in two dialogs (see Figure 28.1). In the **General** dialog sheet, you specify the data medium on which you want to store the backups and how often you want the backups to be run.

Usually, you select **Local Folder** as the location of the backup, and then you can select any directory where you are allowed to read and write files. If you do not want to use an external hard disk or USB flash drive exclusively for backups, it's useful to set up a separate backup directory there. A server that can be accessed via SSH or a cloud directory (such as Google Drive) can also serve as a backup destination. In the past, Déjà Dup used to support

Amazon S3 as well as some other cloud services. However, these functions are now considered obsolete.

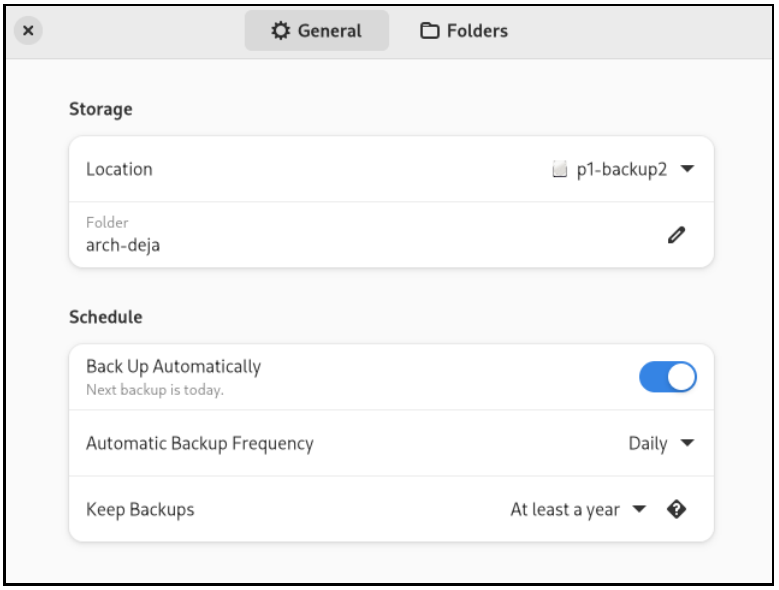


Figure 28.1 Backup Configuration in Déjà Dup

Determining when to back up

If you activate the central **Automatic Backup** switch, you can set the frequency of backups (daily or weekly). In the future, Déjà Dup will always start automatically as soon as you log in and will then take care of running the backups. If the backup medium is not accessible at a given moment, Déjà Dup postpones the backup until later and starts the backup automatically as soon as the external hard disk is accessible. This is extremely convenient: you only need to make sure that a backup media is available on a regular basis, and the backup program will take care of everything else by itself.

Determining how many backup versions to store

Finally, you can specify the period over which the backup program should back up files that change. The default setting of **Forever** is the safest, but it inevitably means that even the largest backup medium will be full at some point. By using the **At Least One Year** or **At Least Three/Six Months** setting, *all* files in the backup will be saved, but in case of changed files only the versions of the last three, six, or twelve months are kept.

Determing what will be backed up

In the **Folder** dialog, you can set which directories you want to be backed up; usually, that's only the home directory.

In **Folders to Ignore**, you can specify the exceptions. By default, the trash can and the `Downloads` directory are already provided there. It's often useful to also exclude directories with very large files from regular backups, such

as videos or the location of virtual machines if you use VirtualBox. (In general, a backup of virtual machines can only work if the virtual machines are not running during the backup.)

The first backup

The **Backup Now** button allows you to start a backup manually at any time. For the first backup, you must also specify whether the backups should be encrypted and, if so, which password should be used. Note, however, that encrypting requires additional CPU power!

The first backup performed using Déjà Dup takes a disproportionately long time. This is because the backup is compressed. For further backups, only the changes are saved, which speeds up the process. Unfortunately, incremental backups are not really fast either, even if only a few files have changed since the last backup.

28.1.2 Restoring Data

You can use the **Restore** button to restore a full backup. For this purpose, you can select the desired backup version and specify where the backup files should be copied.

28.2 Back In Time

Like Déjà Dup, Back In Time is a backup program for personal data. Most current distributions provide ready-made Back In Time packages (package name `backintime-qt`). Only directories accessible locally or via SSH are supported as backup targets.

Benefits and drawbacks

Compared to Déjà Dup, the interface of Back In Time is much more technically oriented. There is no mistaking the KDE origin of the program. During the initial configuration, there are many more options available to influence the details of the backup.

Behind the scenes, Back In Time uses the `rsync` command. During the first backup, a complete backup is created. Later, only the files that have changed since then will be backed up. The connection between the different backup versions is done by hard links. However, this only works on media that support such links. (If you use an external hard disk as a backup medium, you should set up a Linux file system there.) This relatively simple backup technique has two huge advantages (compared to Déjà Dup): For one thing, the backups are created extremely quickly and with low CPU load (at least as long as you do without encryption). Second, the file and directory structure is preserved in the backups. This enables data to be restored both quickly and easily—even without Back In Time, if you want.

Documentation

You can find more information about Back In Time on the following websites:

- *https://backintime.readthedocs.io*
- *https://github.com/bit-team/backintime*

28.2.1 Configuration and Use

Back In Time allows you to set up multiple profiles for different backups. As long as you only need one profile, you can simply edit the **Main Profile** that already exists. The configuration of a profile is done in six dialog sheets (see Figure 28.2):

- **General**
  Here you specify in which directory you want to store the backups. It can also be an external data medium if it is permanently connected to your computer. You must have write permissions for the backup directory.

  You can also set how often the backups should be performed in this dialog sheet. For computers that are not in constant use, you should select the **Repeating (anacron)** variant. In a second step, you can then specify after how many hours, days, or weeks a backup should be automatically performed again.
- **Include**
  Here you select the directories you want to back up. Usually you will simply specify your home directory here. However, you can also make a differentiated selection and, for example, back up only your `Documents` and `Images` directories.
- **Exclude**
  Here you specify which directories and file patterns are excluded from the backup, such as `Downloads` (see Figure 28.2). By default, Back In Time excludes local directories such as `.gvfs` and `.cache`, as well as various system files. (The rules for system files are only relevant if you create a backup of the root directory via Back In Time.) You can add rules for other directories here, such as for `Downloads` or for `.local/share/Trash`.
- **Auto-remove**
  To prevent the backup volume from growing without limits, you can specify which backup data should be deleted automatically. In most cases, it makes sense to activate the **Intelligent Delete** option; this way, you can make sure that backups from the current day and the day before will never be touched. Older backups are gradually deleted, but it is ensured that there is a backup for the preceding seven days, for the preceding four weeks, for the preceding 24 months, and one backup per year.

- **Options and Expert Options**
  Here you will find some options for advanced users, which usually do not need to be changed.

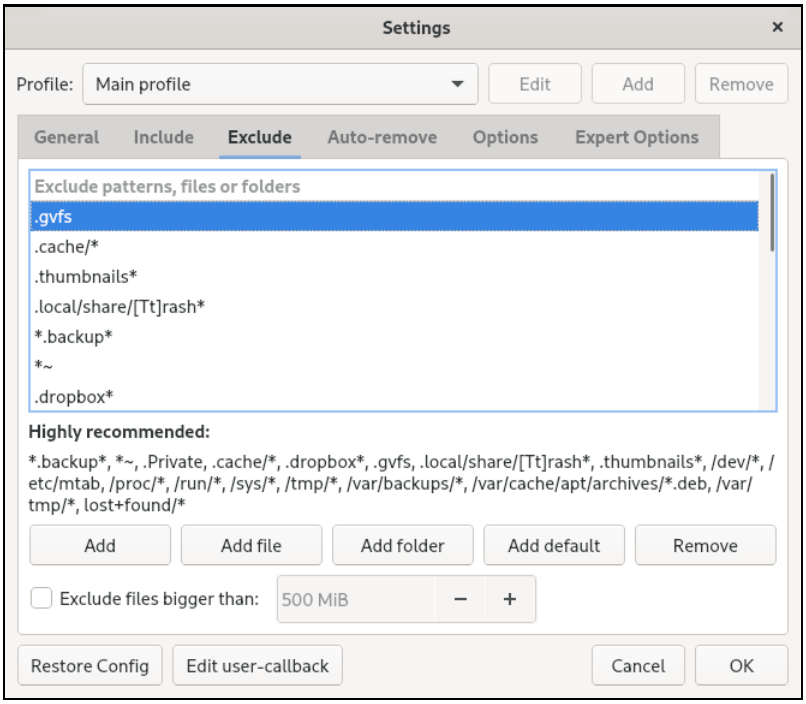


Figure 28.2 Configuration of Back In Time

When the configuration is complete, the Back In Time user interface appears. Here you can start a backup manually at any time, including outside the set backup period. You can also give names to existing backups.

System Backups

By default, Back In Time can only back up personal files. If you want to use Back In Time to back up system files, you must start it in root mode. Both KDE and GNOME provide for corresponding startup entries.

28.2.2 Restoring Data

Once you set up Back In Time, Cron will take care of performing the backups automatically. It analyzes the `.config/backintime/config` configuration file. The Back In Time user interface does not need to be running for the automatic backups! The Cron control is enabled by the `/var/spool/cron/crontabs/loginname` file.

You do not need to restart the Back In Time user interface until you want to make changes to the backup configuration or restore files. There is a directory browser in the program for this purpose (see Figure 28.3). There you select the relevant backup version in the first column, the backed-up directory in the second column (usually **Personal Folder**), and then the file or directory to be restored in the third column. Using the context menu or the **Restore** menu entries, you can then reconstruct the file from the backup.

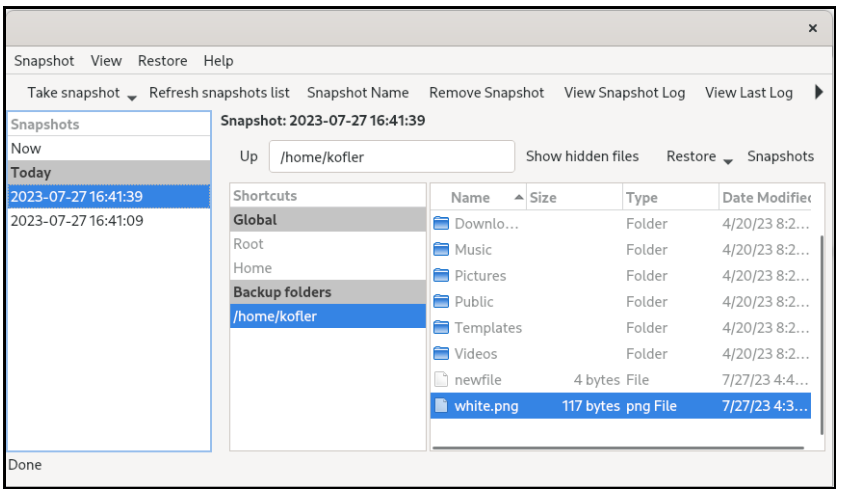


Figure 28.3 Back In Time Browser for Restoring Files

28.3 Grsync

It's an exaggeration to call Grsync a backup tool. In reality, it's a simple user interface for the `rsync` command (see Section 28.6) to synchronize one directory with another.

Benefits and drawbacks

The biggest advantages of Grsync are its ease of use and the fact that your files are copied 1:1 to a second directory. If you ever need to go back to your backup, you don't need any special tools to do so and you can restore the required files quickly and easily. The speed factor in particular should not be underestimated. Restoring data from an incremental backup system with encryption and compression can take hours!

What is annoying, however, is its all-or-nothing approach: Grsync does not provide an easy way to exclude individual subdirectories or specific file types from synchronization. (Experts can specify appropriate `rsync` command options directly in the **Advanced Options** dialog sheet. But then you might as well call `rsync` manually or in a Cron job.) In addition, Grsync lacks many features that are self-evident in "real" backup programs. In particular,

only the most recent version of a file is available in the synchronized directory. If, for example, you have made changes in a text file by mistake and now need an old version of the text, you are out of luck. In this respect, Grsync is a conceivable addition to Déjà Dup; as the only backup tool, it is only useful in exceptional cases.

A possible alternative to Grsync is *Syncthing*. Syncthing is more complex to use, but it offers a lot more configuration options (especially for exception rules).

28.3.1 Configuration and Use

After installation, you connect an external hard disk to your computer, start Grsync, and copy the contents of your home directory to a directory on the external hard disk. All files must be copied the first time, then only changed or new files. You can leave the numerous options essentially as they are preset (see Figure 28.4).

The **Delete on Destination** option is important. It specifies whether Grsync should also synchronize delete operations. If you delete a file in your home directory after the first backup, this file will also be deleted in the backup directory the next time. If you want your backup to be protected against accidental deletions, you must not enable this option, which is the basic setting anyway. If, on the other hand, it is important to you that the backup has exactly the same content as the directory to be backed up, you should enable the option.

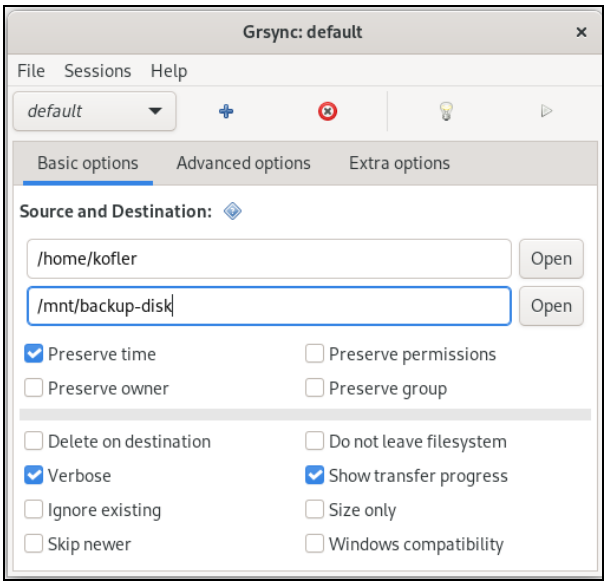


Figure 28.4 Synchronizing Directories Using Grsync

28.4 Borg Backup

The backup programs presented so far are all equipped with a graphical user interface and are primarily aimed at desktop users. But servers also need backups. And advanced Linux users prefer tools to be called as commands even on the desktop.

Before I present low-level tools for programming your own backup scripts starting in the next section, I want to present Borg Backup here (*https://borgbackup.org*). In terms of its functions, this is a command that is in a league with Déjà Dup.

Benefits and drawbacks

Borg Backup creates incremental, encrypted, and compressed backups, either in a local directory or on another server where Borg Backup is also installed. (This means that Borg Backup is incompatible with cloud storage such as Amazon S3. After all, some server providers now offer Borg functions for their backup space.) Borg Backup avoids storing data multiple times by deduplicating at the block level. As with Déjà Dup, this results in backup files from which you can restore your files exclusively through Borg Backup. This black box concept is characteristic of modern backup solutions, but also gives me headaches with command-oriented tools. On the other hand, a mature backup program used by many users is usually more robust and secure than any homemade solution.

Of course, Borg Backup as a command with countless options is less accessible for beginners than a backup solution with a user interface. However, Borg Backup must be given credit for the fact that both the operation of the program and the underlying techniques are extremely well-documented. Another plus point: Borg Backup is programmed in Python, so it uses libraries that are widely used on Linux.

28.4.1 Installation

Borg Backup is so popular in the server scene that ready-to-use packages are available for almost all common Linux distributions. This facilitates its installation:

```
root# apt/dnf/zypper install borgbackup
```

28.4.2 Usage

Setting up a backup repository

Borg Backup is controlled by the `borg` command. Before you can perform your first backup, you need to create a backup repository, which is the location for the backup. In the following example, this is a local directory on an external hard disk/SSD:

```
user$ borg init --encryption=repokey \
        /media/kofler/p1-backup/borgtest/
Enter new passphrase: **********
Enter same passphrase again: **********
```

When you run this command, `borg` generates a key that is contained in the backup directory in the `config` file. The access to the key is secured by a passphrase. In the future, this passphrase must be specified each time the backup is accessed.

Running backups

The following command creates a first backup of the `data/linux` directory. Each backup must have a name that is appended as `::name` to the repo directory:

```
user$ borg create --stats \
        /media/kofler/p1-backup/borgtest/::initialbackup \
        data/linux
Enter passphrase for key /media/kofler/p1-backup/borgtest:
**********
Archive name: initialbackup
Archive fingerprint: 3e5b6528...

Time (start): 2023-05-14 09:34:24
Time (end):   2023-05-14 09:34:44
Duration: 19.86 seconds
Number of files: 3329
Utilization of max. archive size: 0%
                Original size      Compressed size    Deduplicated size
This archive:       1.42 GB            848.97 MB            790.39 MB
...
```

As expected, the second backup with few changes is much faster and requires comparatively little additional space due to deduplication:

```
user$ borg create --stats \
        /media/kofler/p1-backup/borgtest/::secondbackup \
        data/linux
Enter passphrase for key /media/kofler/p1-backup/borgtest:
**********
...

Duration: 1.26 seconds
                Original size      Compressed size    Deduplicated size
This archive:       1.42 GB            848.93 MB              2.21 MB
All archives:       2.84 GB              1.70 GB            792.60 MB
```

Viewing backups

`borg list` displays all backups available in a repository or the files contained in an explicitly specified backup:

```
user$ borg list /media/kofler/p1-backup/borgtest/
initialbackup  2023-05-14 09:34:24 [3e5b6528...]
secondbackup   2023-05-14 09:40:04 [78f44df7...]
user$ borg list /media/kofler/p1-backup/borgtest/::secondbackup
drwxr-xr-x  ...      0 2023-05-13 21:58:16 data/linux
-rw-r--r--  ...   3924 2023-04-27 13:40:05 data/linux/header-keys.tex
-rw-r--r--  ...  44798 2023-05-09 15:11:53 data/linux/header.tex
-rw-r--r--  ...   4559 2023-05-13 08:24:36 data/linux/book.tex
...
```

Restoring data

To unpack a complete backup in the currently active directory, you want to call `borg extract`. Existing files and directories are overwritten without confirmation. If you want visual feedback when unpacking, you need to specify the `--progress` or `--list` option:

```
user$ borg extract /media/kofler/p1-backup/borgtest/::secondbackup
```

It's also possible to extract only individual files:

```
user$ borg extract /media/kofler/p1-backup/borgtest/::secondbackup \
        data/linux2023/text/apps.tex data/linux2023/text/backup.tex
```

Deleting backups

To delete a backup from a repository, you must call `borg delete`:

```
user$ borg delete /media/kofler/p1-backup/borgtest/::backup123
```

`borg prune` is usually more useful. This command allows you to delete older backups in such a way that a certain number of older backups is preserved. In the following example, these are one backup each for the last seven days, for the last four weeks, and for the last 24 months:

```
user$ borg prune --keep-daily 7 --keep-weekly 4 --keep-monthly 24 --
list \
        /media/kofler/p1-backup/borgtest/
```

28.4.3 Borg Backup in Scripts

Borg Backup is designed to automate backups. A script to perform a regular backup can be structured as shown in the following listing:

```
#!/bin/bash
export BORG_REPO='/media/kofler/p1-backup/borgtest/'
export BORG_PASSPHRASE='topsecret'
```

```
cd /crypt/home/kofler
borg create --stats ::'{now}' data/linux2023
borg prune  --list --keep-daily 7 --keep-weekly 4 --keep-monthly 12
```

The BORG_REPO and BORG_PASSPHRASE variables are automatically analyzed by borg and specify the backup repository and the corresponding password. The script uses the current date and time as the backup name with ::'{now}'. Other ways to pass passwords are described at *https://borgbackup.readthedocs.io/en/stable/faq.html#how-can-i-specify-the-encryption-passphrase-programmatically*.

28.4.4 Borg Backup Using SSH

Borg Backup works quite straightforwardly via an SSH connection. The only requirement is that a sufficiently up-to-date version of Borg Backup is also installed on the server:

```
user$ borg init --encryption=repokey user@remotehost:path/to/repo
Enter new passphrase: **********
Enter same passphrase again: **********
user$ borg create --stats \
  user@remotehost:path/to/repo::initial local/data
Enter passphrase for key ssh://user@remotehost/./path/to/
repo: **********
```

28.4.5 Internal Details

Configuration files When setting up a new repository, Borg Backup stores some information in .config/borg in the home directory, including the location of the repository. Note that no copy of the backup key is saved. In addition, Borg Backup takes up space in the .cache/borg directory for various caches to make backups more efficient.

Compression Borg Backup uses the lz4 compression method by default. The compression is worse than with gzip, but fast and with low CPU load. Borg Backup also supports other compression methods (--compression option; see also borg help compression). You can mix different compression methods in one repository. The --compression option applies only to newly added data blocks.

28.4.6 Borg User Interfaces

Borg Backup is actually intended for use as a command. However, this has not stopped the open-source community from developing wrappers or

user interfaces to make the program easier to use. At present, however, it isn't yet possible to assess whether any of these programs will be able to establish themselves in the longer term:

- *https://github.com/borgmatic-collective/borgmatic* (simplified command)
- *https://github.com/KenKundert/emborg* (simplified command)
- *https://github.com/borgbase/vorta* (real GUI)

In particular, the Vorta interface developed in Python looks promising. The program can be installed using the `pip3` Python package manager:

```
user$ pip3 install vorta
user$ vorta
```

28.5 Compressing and Archiving Files

Now that I've introduced you to complete backup tools in the first sections of this chapter, I will now present various low-level commands that will help you archive and back up files in different ways: `tar`, `zip`, `rsync`, `rdiff-backup`, `rsnapshot`, and more. These commands enable you to program your own backup scripts. For examples of such scripts, see Section 28.9 and Section 28.10.

At this point, I want to start with commands for compressing and archiving files. An initial overview is provided in Table 28.1.

Command	Meaning
`gzip`	Compresses a file
`gunzip`	Decompresses the file
`bzip2`	Compresses a file (higher compression than `gzip`, slower)
`bunzip2`	Decompresses the file
`xz`	Compresses a file (higher compression than `bzip2`, slower)
`unxz`	Decompresses the file
`lzop`	Compresses/decompresses much faster than `gzip`
`tar`	Creates or extracts a file archive
`zip`	Creates a Windows-compatible ZIP archive
`unzip`	Extracts a ZIP archive
`zipinfo`	Displays information about a ZIP archive

Table 28.1 Tools for Compressing and Archiving Files

28.5.1 Compressing Files (gzip, bzip2, xz, and lzop)

gzip and gunzip

`gzip` compresses the files specified as parameters and renames them to `name.gz`. `gunzip` works in the opposite direction. The two commands use the LZ77-Lempel-Ziv algorithm, which is particularly suitable for text files, but not for audio or video files. Compression is of course *lossless*; that is, the original file is available again unchanged after decompression. The following commands demonstrate its use:

```
user$ ls -l filesystem.tex
... 178794  1. Aug 17:43 filesystem.tex
user$ gzip filesystem.tex
user$ ls -l filesystem.tex.gz
...  57937  1. Aug 17:43 filesystem.tex.gz
user$ gunzip filesystem.tex.gz
```

bzip2 and bunzip2

`bzip2` and `bunzip2` represent alternatives to `gzip`/`gunzip`. The advantage of these commands is their slightly better compression; the disadvantage is their slightly slower execution. The file extension of files compressed in this way is `.bz2`:

```
user$ bzip2 filesystem.tex
user$ ls -l filesystem.tex.bz2
...  47105  1. Aug 17:43 filesystem.tex.bz2
user$ bunzip2 filesystem.tex.bz2
```

xz, unxz, and 7zr

Instead of `gzip` and `bzip2`, you can also use `xz` for compression. In most cases, the result is even smaller files, but the compression requires even more time or CPU resources. If the smallest possible compression is the primary goal, you can also try the `7zr` command from the `p7zip` package.

lzop

The objective of `lzop` is quite different: this compression command works *much* faster than all the commands mentioned so far, but the resulting files are comparatively large (in my tests they were about 50% larger than with `gzip`). The use of `lzop` is especially recommended if you want to compress on the fly with as little CPU load as possible—for example, to transfer a large file over a network connection.

In the following sample command, a logical volume is read using `cat` and compressed using `lzop`. This takes only slightly longer than copying the logical volume directly to an image file:

```
root# cat /dev/vg1/lv3 | lzop -c > lv3.img.lzo   (55 seconds)
root# cat /dev/vg1/lv3 > lv3.img                 (50 seconds)
```

28.5.2 Creating Compressed Archives (tar and zip)

tar

`tar` is the preferred command to combine multiple files into one archive on Linux, usually compressing the archive by using `gzip` or `bzip2`. `tar` was originally designed to write files to or read files from a streamer. Because such streamers are only used relatively rarely, I will only describe their use for file archives at this point.

The following command inserts all files from the `book` directory into the `myarchive.tgz` compressed archive file. But before that, let me briefly explain the option letters: `c` stands for *create*; `tar` is supposed to create an archive. `z` stands for *zip*; the archive is to be compressed using `gzip`. `f` stands for *file*; `tar` is supposed to create an archive file instead of writing the archive to a streamer cartridge. You can specify the file name you want to use after the option. The usual file identifier for such archives is `.tar.gz` or `.tgz` for short:

```
user$ tar -czf myarchive.tgz book/
```

`tar -tzf` returns a table of contents of the archive. The files within the archive are ordered arbitrarily. In most distributions, `less` is configured in such a way that you can simply view the archive contents via `less name.tgz`:

```
user$ tar -tzf myarchive.tgz
linuxbook/
linuxbook/lanserver.tex
linuxbook/security.tex~

linuxbook/book.tex
linuxbook/c4.txt~
...
```

`tar -xzf` unpacks the archive and extracts all contained files:

```
user$ cd other-directory/
user$ tar -xzf myarchive.tgz
```

In the following example, `tar` extracts only `*.tex` files from the archive. Pay attention to the apostrophes for the file pattern to avoid an immediate analysis by the shell:

```
user$ tar -xzf myarchive.tgz '*.tex'
```

If you want to compress archives using `bzip2` instead of `gzip`, you should replace the `z` option with `j` (i.e., `tar -xjf ...`).

zip In the Unix/Linux world, TAR files are the preferred format for distributing file archives. However, if you communicate with Windows users, ZIP archives are the better choice. The following command inserts all HTML files passed as parameters into `myarchive.zip`:

```
user$ zip myarchive.zip *.html
```

If you want to archive the contents of entire directories, you need to specify the `-r` option:

```
user$ zip -r myarchive.zip mywebsite/
```

zipinfo You can view the contents of a ZIP file via `zipinfo`:

```
user$ zipinfo myarchive.zip
Archive:  test.zip   143677915 bytes   1899 files
2.3 unx    78039 tx defN 10-Jul-16 11:27 linuxbook/lanserver.tex
2.3 unx   115618 tx defN  7-Apr-15 15:58 linuxbook/security.tex
2.3 unx     3899 tx defN 28-Jul-16 16:38 linuxbook/book.tex
2.3 unx      752 tx defN 11-Feb-14 12:06 linuxbook/c4.txt
...
```

unzip To extract the archive, you can use `unzip`:

```
user$ cd other-directory/
user$ unzip myarchive.zip
```

28.6 Synchronizing Directories (rsync)

The `rsync` command synchronizes directory trees. You can use it to copy all files from one directory to a new directory in the first pass. In subsequent runs, only changed files will be copied and (if you want) deletions will also be replicated. `rsync` is therefore ideal for synchronizing a complete copy of a directory on an external hard disk on a daily or weekly basis, for example. `rsync` can also be used via network connections.

By default, the communication takes place via `ssh`. This means that the data transfer is also encrypted at the same time. As an alternative, you can use another external shell program or configure an `rsync` server on the remote side.

Table 28.2 shows the syntax for specifying the source and destination directories. Table 28.3 summarizes the most important options for controlling `rsync`.

Syntax	Meaning
`file1 file2`	Local files
`directory`	Local directory
`host:dir`	Directory on the `host` computer
`user@host:dir`	As above with SSH login under the name `user`
`rsync://user@host/dir`	Communication with the `rsync` server
`rsync://user@host:port/dir`	`rsync` server on the specified port

Table 28.2 Specification of Source and Destination Directories in rsync

Option	Effect
`-a` or `--archive`	Copies recursively and gets all file information
`--delete`	Deletes files or directories in the target directory that no longer exist in the source directory
`-D`	Also takes into account device and special files
`--exclude=samples`	Skips the specified files
`-g` or `--group`	Receives the group membership
`-l` or `--links`	Duplicates symbolic links
`-o` or `--owner`	Receives the owner information
`-p` or `--perms`	Receives the access permissions
`-r` or `--recursive`	Recursively copies all subdirectories as well.
`-t` or `--times`	Receives the modification time
`-u` or `--update`	Ignores already existing older files
`-v` or `--verbose`	Indicates what is currently happening
`-W` or `--whole-file`	Copies the entire file when changes have been made
`-z`	Compresses the data transferred via SSH

Table 28.3 rsync Options

Local use To synchronize an entire directory, including all subdirectories, you can use the `-a` option, which is the short notation for a whole bunch of other options (`-rlptgoD`). This option causes a recursive processing of all subdirectories and ensures that as much file information as possible is retained: owner, group membership, time of last change, and so on. If `dir2` doesn't exist yet, the directory will be created. In contrast to `cp`, existing files that have remained unchanged since the last copy will not be copied again:

```
user$ rsync -a dir1/ dir2/
```

By default, `rsync` copies or updates all new or changed files, but does not delete anything. If you want files or directories deleted from `dir1` to also be deleted in `dir2`, you must also specify the `--delete` option. It should be clear that this option is dangerous: if you delete a directory by mistake, this very directory will also be deleted on the backup hard disk during the next backup operation!

Network use When using `rsync` to synchronize directories on different computers, you must specify the source and destination directories in the `hostname:directory` or `username@hostname:directory` notation, respectively. In the first case, `rsync` uses the current user name.

The following command synchronizes the `dir1` directory of the local user `username` on the `saturn.sol` computer with the `dir2` directory on the `mars.sol` computer. The password entry is the responsibility of `ssh`. You must therefore enter the login password of the `username` user on the `mars.sol` computer:

```
username@saturn.sol$ rsync -e ssh -az dir1/ mars.sol:dir2/
username@mars.sol's password: ******
```

`rsync` can also transfer files from a remote computer to the local one. Thus, the following command synchronizes in the opposite direction:

```
username@saturn.sol$ rsync -e ssh -az mars.sol:dir2/ dir3/
username@mars.sol's password: ******
```

Of course, if `rsync` is to be called by an automatic backup script, the interactive password entry will interfere. The solution here is to set up a private key file on the local computer and the matching public key on the partner computer (see Chapter 23, Section 23.4). If you omit the *passphrase* when generating the keys, SSH login will be possible without a password. For security reasons, you should provide a separate account for the backup script.

rsync server Instead of using SSH directly, the local `rsync` command can communicate with an `rsync` server on the remote computer. This assumes that an `rsync`

server has been set up on the partner computer. Its configuration is carried out by the `/etc/rsyncd.conf` file.

The communication between the `rsync` client and the `rsync` server takes place via port 873, which must not be blocked by a firewall.

In real life, the configuration of an `rsync` server is usually only recommended if the server is regularly mirrored by different clients—that is, if it serves as a mirror server:

- *https://unix.stackexchange.com/questions/26182*
- *https://serverfault.com/questions/100707*

rclone and Celeste

`rsync` expects an SSH server on external servers. However, this requirement is not met for typical cloud services. In such cases, you may consider using `rclone` or its graphical user interface Celeste (currently in alpha testing):

- *https://rclone.org*
- *https://github.com/hwittenborn/celeste*

28.7 Incremental Backups (rdiff-backup)

An interesting alternative to `rsync` is the `rdiff-backup` command. The most important difference from `rsync` is that when files are changed, `rdiff-backup` also archives the old version in the backup directory. To save space, instead of saving a copy of the file in question, only the changes can be saved, optionally in compressed form. `rdiff-backup` thus provides an incremental backup without much effort, from which you can also restore older versions of a file. Basically, `rdiff-backup` provides the same functions as Apple's Time Machine on macOS—only without a spectacular user interface.

Running backups

In its simplest form, you apply `rdiff-backup` to two local directories. If the repository directory doesn't exist yet, it will be created:

```
root# rdiff-backup /home /home-backup
```

`rdiff-backup` creates the `rdiff-backup-data` subdirectory in the backup directory. In it, it stores various statistical data and status information. In addition, the `increments` directory contains old versions of files that have since changed or that have been deleted. Only the changes are saved (`.diff`) and additionally compressed. In addition, the date of the last version is integrated into the filename, which results in confusing filenames like `filename.2023-11-03T08:37:58+02:00.diff.gz`.

Access to backups

If you want to access the backup, `/home-backup` contains the state of the `/home` directory at the time of the last backup, except for the `rdiff-backup-data` directory, exactly as if you had run the backup using `cp -a` or `rsync -a --delete`. As a result, accessing the last backup is quite simple. Of course, you can also restore the backup using `rdiff-backup`. To do this, you need to use the `-r` option and the time specification `now`. The following command restores the backup to a temporary directory on a trial basis:

```
root# rdiff-backup -r now /home-backhup /tmp/home-current
```

If you want to access an older version of a file or a file that has since been deleted, things get more complicated: you need to apply all `.diff` files in order (the latest one first) until you have restored the relevant time in the past. Of course, you don't have to do this manually; `rdiff-backup` will help you. The following command restores the state of the `/home` directory to the way it was 10 days previously:

```
root# rdiff-backup -r 10D /home-backup/ /tmp/home-historical
```

You can specify the backup time either in absolute terms (such as `2023-12-31`) or relatively in hours (`h`), days (`D`), weeks (`W`), and so on; see also `man rdiff-backup` in the `TIME FORMATS` section. Note that restoring old files with an increasing number of versions requires a considerable amount of CPU and is correspondingly slow!

Often you only want to restore a single file or a subdirectory in an old version. You can also specify a file that no longer exists or a directory that has been deleted in the meantime:

```
root# rdiff-backup -r 10D /home-backup/file  file-historical
root# rdiff-backup -r 10D /home-backup/dir/  dir-historical
```

Deleting old backups

If you run `rdiff-backup` regularly, the backup directory will grow bigger and bigger over time. To delete all backup files older than four months, you should proceed as follows:

```
root# rdiff-backup --remove-older-than 4M --force /home-backup/
```

Instead of a specific time, you can also specify the maximum number of backup versions that should remain archived. The following command reduces the backup versions to three:

```
root# rdiff-backup --remove-older-than 3B --force /home-backup/
```

Network backup

In all the examples so far, I have assumed that the source and destination directories are in the local file system. `rdiff-backup`, however, can also access external directories over the network. Unlike `rsync`, `rdiff-backup`

must also be installed on the external computer! Communication with the platform occurs via web services, and no special configuration of `rdiff-backup` is required.

When you specify external directories, almost the same syntax applies as for `rsync`. The only difference is that *two* colons must be specified after the host name:

```
root# rdiff-backup user@company-abc.com::/home /home-backup
```

You can find even more details and examples of how to use `rdiff-backup` at *https://rdiff-backup.net*.

duplicity

If the `rdiff-backup` concept appeals to you, but you also want to have the backup encrypted and uploaded via SSH or FTP to an external server, it might be worth taking a look at the Python program called `duplicity` (*https://duplicity.gitlab.io*). This program is similar to `rdiff-backup`, but it creates `tar` archives. The command serves as the basis for the Déjà Dup backup user interface, which I introduced at the beginning of this chapter.

28.8 Incremental Backups (rsnapshot)

The `rsnapshot` Perl script from the package of the same name also builds on `rsync`. Unlike the `rdiff-backup` command just described, it uses hard links to access previously backed up files from previous backups. This makes accessing older backup versions (snapshots) easier than when you use `rdiff-backup`. A compression of the backups, on the other hand, is not provided.

`rsnapshot` is designed to run the script automatically on a regular basis. `rsnapshot` backs up both local directories and, via SSH, directories from other machines on the local network when configured appropriately. All backups are stored on the local machine running `rsnapshot`. This approach is exactly the opposite of many other backup tools: `rsnapshot` is not intended to save a backup of local files on another computer. Rather, the command backs up the data of multiple computers that are accessible via `ssh` or `rsync` to the local file system.

Configuration

Instead of having to pass numerous options to `rsnapshot`, you can control the command through the `/etc/rsnapshot.conf` configuration file. Two important syntax rules apply to this file: directories must end with a slash (i.e., `/directory/`, not `/directory`), and the elements of the configuration file must be separated by tab characters (not spaces).

For first experiments, you can leave the configuration file provided together with `rsnapshot` unchanged except for a few details. The three most

important parameters you need to know and set yourself if necessary are snapshot_root, backup, and interval.

snapshot_root snapshot_root specifies the directory in which you want to store the backups. By default, the /var/cache/rsnapshot directory is used, which was automatically set up during the installation of rsnapshot.

backup backup specifies which directory is to be backed up where. backup can be specified multiple times, each in a separate line, to back up multiple directories to different locations. For backups within the local file system, the backup settings look as follows:

```
# in /etc/rsnapshot.conf
...
backup  /home/   localhost/
backup  /etc/    localhost/
```

To back up a /home/user/mail directory of the external mars.sol host that is accessible via SSH, you need to remove the comment character (#) in the already existing cmd_ssh line. In the backup line, you want to specify the login name, the host name, and the entire directory to be backed up. For this to work, you must copy an SSH key set up with root privileges without a passphrase to the /home/user/.ssh/authorized_keys file of the mars.sol computer prior to the first backup (see Chapter 23, Section 23.4):

```
# in /etc/rsnapshot.conf
...
cmd_ssh /usr/bin/ssh
...
backup  user@mars.sol:/home/user/Mail/  mars.sol/
```

Caution: The key file stored on the local computer in the /root/.ssh directory must not fall into the wrong hands!

Optionally, it's possible to create an LVM snapshot or run scripts during the backup, exclude certain file patterns from the backup, and so on. Details can be found in the supplied configuration file as well as on the man page for rsnapshot.

interval interval defines how many backup versions should be stored for a given time interval. The default settings look as follows:

```
# in /etc/rsnapshot.conf
...
interval        hourly  6
```

```
interval        daily   7
interval        weekly  4
#interval       monthly 3
```

This means that six versions of backups run by the `rsnapshot hourly` command are stored. Furthermore, seven backups of `rsnapshot daily` are archived as well as four backups of `rsnapshot weekly`. The `monthly` interval is not defined by default.

The `interval hourly 6` setting is useful if the `rsnapshot hourly` command is not run every hour, but only every four hours, as provided in `/etc/cron.d/rsnapshot`. On the other hand, if you really want to run `rsnapshot hourly` every hour, the `interval hourly 24` setting would be more appropriate.

You can define any additional intervals as needed. `rsnapshot` stores backups for each interval in a separate directory and links unchanged backups only within the directory of an interval. This means that the memory requirement increases significantly with each additional interval.

Manual call

`rsnapshot` can be called manually, but this requires `root` privileges. You must specify a time interval as a parameter:

```
root# rsnapshot daily
```

After the backup, you will find the backed up data in the following directory:

```
/var/cache/rsnapshot/<interval.n>/<hostname>/<directory>
```

Here, `interval` specifies the interval: `hourly` by default, `daily`, `weekly`, or `monthly`. `n` specifies the backup version: 0 for the most recent backup, 1 for the latest version, 2 for the version before last, and so on. `hostname` specifies the host from which the backed-up data originates, where `localhost` denotes the local host. Thus, the latest hourly backup of the local `/etc` directory is contained in the following directory:

```
/var/cache/rsnapshot/hourly.0/localhost/etc
```

The most recent monthly backup of the `/home/user/Mail` directory of the `mars.sol` server is located in the following directory:

```
/var/cache/rsnapshot/monthly.0/mars.sol/home/user/Mail
```

Automatic call

To automate the backups, a regular call of `rsnapshot` is provided by the `/etc/cron.d/rsnapshot` Cron configuration file. All you have to do is remove the comment lines before the four cron lines already provided:

```
# /etc/cron.d/rsnapshot
0 */4         * * *           root    /usr/bin/rsnapshot hourly
30 3          * * *           root    /usr/bin/rsnapshot daily
0  3          * * 1           root    /usr/bin/rsnapshot weekly
30 2          1 * *           root    /usr/bin/rsnapshot monthly
```

This will run `rsnapshot` every four hours—at 0:00, 4:00, 8:00, and so on; daily at 3:30; weekly on Mondays at 3:00; and monthly on the first day of the month at 2:30, with `hourly`, `daily`, `weekly`, and `monthly` parameters. Of course, you can vary the times as you see fit. Note that all time intervals used in the Cron file must also be defined in `/etc/rsnapshot.conf`! This is not the case for the `monthly` interval by default.

28.9 Backup Scripts

Automating rsync using Cron

This section focuses on real-life examples. The following script synchronizes the contents of the local `data` directory with the directory of the same name on an external hard disk called `backup`. The script tests whether the external hard disk is available. If it isn't, the backup will not be performed:

```
#!/bin/bash
if [ -d /media/backup/ ]; then
  rsync -avW --delete /home/kofler/data /media/backup/
fi
```

As a rule, you will run backup scripts automatically on a regular basis. The easiest way to do this is to set up a Cron job. To do so, it's easiest to save the script directly in a Cron directory—for example, under the `/etc/cron.daily/mybackup` filename. Please note that the filename must not contain a dot and that the file must be executable (`chmod a+x`).

The other variant is to save the backup script in any directory—for example, `/etc/myscripts`. To call the script automatically, you need to set up a new file in `/etc/cron.d`, such as according to the following pattern:

```
# file /etc/cron.d/mybackup
15 2 * * * root /etc/myscripts/mybackup
```

Thus, the `mybackup` backup script will run daily at 2:15.

Daily and monthly backups

The following script creates a compressed TAR archive of the `data` directory. The archive is saved under two file names: `mydata-day-DD.tar.gz` and `mydata-month-MM.tar.gz`. `DD` indicates the day (01 to 31) and `MM` the month (01 to 12). If the script is run daily, you will have 43 backup versions over time,

reflecting the state of the backup directory for the last 28 to 31 days and for the last 12 months:

```
#!/bin/bash
fname1=/backup/mydata-day-$(date "+%d").tar.gz
fname2=/backup/mydata-month-$(date "+%m").tar.gz
tar czf $fname1 /home/kofler/data
cp $fname1 $fname2
chmod 600 $fname1 $fname2
```

Backups using LVM

If files change while a backup is being performed, the resulting backup will be inconsistent. However, it isn't always possible to stop the programs or server services in question for the backup. One possible solution to this problem is to create an LVM snapshot at the beginning of the backup and use it as the basis for the backup. Of course, this assumes that the directory with the data to be backed up is located in a file system that is stored in a logical volume (and not directly on a hard disk partition).

To create a snapshot, you can use the `lvcreate` command with the `-s` option. Use `-L` to specify the maximum amount of data that may change during the lifetime of the snapshot—that is, when the backup is being performed. There must be sufficient free space for this in the LVM storage pool (i.e., in the PV). Initially, the required size of the buffer memory is difficult to estimate. When you are done with the backup, you can use `lvdisplay` to see what percentage of the buffer has been used so far.

In the following sample script, I assume that the `/home` directory is located in the `/dev/vg1/myhome` LV. To perform a consistent backup from the `/home` directory during which no files change, you can use `lvcreate` to create the `homesnap` snapshot. You provide 2 GiB as buffer memory. Then you mount this snapshot at the `/var/homesnap` directory in the directory tree and use it as a data source for your backup command or script. Finally, you need to remove `homesnap` from the file system again and remove the snapshot by using `lvremove`. The `-f` option prevents you from being asked whether you really want this:

```
#!/bin/bash
mkdir -p /var/homesnap
lvcreate -s -L 2G -n homesnap /dev/vg1/home
mount -t ext4 /dev/vg1/homesnap /var/homesnap
tar czf /backup/mybackup.tgz /var/homesnap

lvdisplay /dev/vg1/homesnap > /tmp/backup.log
umount /var/homesnap
lvremove -f /dev/vg1/homesnap
```

The `lvdisplay` command is used to check whether you have correctly sized the buffer memory for the LVM snapshot:

```
root# cat /var/homesnap
  ...
  COW-table size         2.00 GB
  Allocated to snapshot  5.23%
```

Thus, in the preceding example, only 5% of the buffer memory was used. Of course, that's no guarantee that it will be the same next time; maybe some user or process is causing big changes in the `/home` directory.

Reading a logical volume as an image

Even if you use logical volumes to store a virtual machine's disk in them (see Chapter 32), you can use LVM snapshots to safely copy the logical volume to a compressed image file. The following script first renames the possibly already existing `image.lzo` backup to `old-image.lzo`. It then creates the `snap` snapshot of the `/dev/vg1/lv1` logical volume that contains the virtual disk. Then the image is loaded via `cat` and compressed using `lzop`. Thanks to `ionice -c 3`, this IO- and CPU-intensive command can be executed without too much affecting all other running processes. The compressed image is then uploaded to a backup server using `curl`. The `--limit-rate` option limits the transfer rate. This makes sure that the server's network capacity is not completely blocked by the backup script:

```
#!/bin/bash
mv /backup/image.lzo /backup/old-image.lzo
lvcreate -s -L 2G -n snap /dev/vg1/lv1
ionice -c 3 cat /dev/vg1/snap | lzop -c > /backup/image.lzo
lvremove -f /dev/vg1/snap

# upload image via FTP to a backup server
pw=u12345:2n34jkj546wqdsr
ftp=u12345.backup-server.de
curl --limit-rate 8m -T /backup/image.lzo -u $pw ftp://$ftp/
image.lzo
```

28.10 Backups to S3 Storage

S3 stands for *Simple Storage Service* and is one piece of the Amazon Web Services (AWS) jigsaw puzzle. S3 is a comparatively inexpensive cloud service from Amazon for storing files. It's particularly popular as external backup storage, especially when you need to perform a backup that's geographically separate from the server in addition to a local backup. This gives

you the peace of mind that you'll be able to access your data even if your hosting company goes out of business or its data center is destroyed by a natural disaster.

The pricing model for Amazon S3 is confusing in that you pay not only for the amount of data stored as such, but also for transfers. S3 even distinguishes between GET and PUT requests (i.e., downloads and uploads). However, don't be confused by the confusing presentation of the pricing model: compared to services like Dropbox, S3 is downright cheap when used reasonably. However, when programming backup scripts, you should try to minimize unnecessary transfers.

28.10.1 Setting Up S3 Storage

Buckets

You can try Amazon S3 with data amounts up to 5 GiB for one year free of charge. However, you must also register for the test with your name, address, a working phone number, and a credit card. It only takes a few minutes to set up your account. After that, you need to create at least one so-called bucket in the S3 web interface, which is a basic directory within S3, so to speak. The bucket name must be unique within the entire S3 cloud. For this reason, it's recommended to include your company or domain name—for example, `mycompany.first.test` or `info.kofler.backup`.

> **Avoid Public Buckets**
>
> In the past, it has happened time and again that business-critical data on Amazon S3 has been made publicly available for the entire world to download. This was due to the bucket being configured too generously. Make sure that the bucket is not public. That being said, it is a good idea to only upload files to the cloud that you have encrypted yourself.

Installing the awscli package

After these somewhat lengthy preparations, you can finally get started in the terminal. Amazon provides the setup files for `awscli` version 2 itself. For this purpose, you want to run the following commands:

```
root# curl https://awscli.amazonaws.com/awscli-exe-linux-x86_64.zip \
        -o awscliv2.zip
user$ unzip awscliv2.zip
user$ less ./aws/install   (take a look at the setup script)
user$ sudo ./aws/install
```

aws configuration

The connection to S3 is encrypted. Before you can configure the `aws` command for further use, you must set up a user and group on the AWS Identity

and Access Management (IAM) page. The group should be linked to the AmazonS3FullAccess policy and to the new user.

You can find the IAM page at *https://console.aws.amazon.com/iam*. There you create an access key for the user and remember the corresponding secret access key, which is displayed only once. Then you must run aws configure in the terminal, specifying the two key strings. aws stores this data and the other options in .aws/credentials and .aws/config:

```
root# aws configure
AWS Access Key ID [None]:     AKxxxxxxx
AWS Secret Access Key [None]: xxxxxxxxxxxxxxx
Default region name [None]:   eu-central-1
Default output format [None]:
```

The correct specification of the region is not that easy. When creating new buckets, you can choose where they should be physically stored. For European S3 customers, for example, Frankfurt is usually the best choice. But how does S3 expect you to specify the region? At this point, it's best to leave the region empty. Later, you can use aws s3api get-bucket-location to determine the region of your buckets and then rerun aws configure:

```
root# aws s3api get-bucket-location --bucket mycompany.first.test
  "LocationConstraint": "eu-central-1"
```

28.10.2 The aws Command

aws s3 ls

aws s3 ls returns a list of all of your buckets, including the date each bucket was created:

```
root# aws s3 ls
2023-05-04 12:49:03 mycompany.first.test
2023-08-06 07:58:11 linux.book.test
```

aws s3 cp

You can use aws s3 cp to upload a local file. The following lines first create a compressed backup of the /etc directory and then transfer it to a bucket specified in the following format: s3://bucketname. Of course, a transfer in the opposite direction is also possible (third command):

```
root# tar czf etc.tgz /etc
root# aws s3 cp etc.tgz s3://linux.book.test
  upload: ./etc.tgz to s3://linux.book.test/etc.tgz

root# aws s3 cp s3://linux.book.test/etc.tgz etc-copy.tgz
  download: s3://linux.book.test/etc.tgz to ./etc-copy.tgz
```

The handling of directories is a little confusing. Officially, there are no directories within an S3 bucket. Rather, all files are located directly in the bucket without a hierarchical order. That is why commands like `mkdir`, `rmdir`, and `cd` are missing.

However, bucket files may have names such as `d1/d2/name`. `ls` processes such filenames as if they were directories. The S3 web interface also splits such filenames into virtual directories:

```
root# touch tst.txt
root# aws s3 cp tst.txt s3://linux.book.test/d1/d2/tst.txt

root# aws s3 ls s3://linux.book.test --recursive
  2023-05-06 08:12:47    2595826 etc.tgz
  2023-05-06 08:30:23          0 d1/d2/tst.txt

root# aws s3 ls s3://linux.book.test/d1/d2/
  2023-05-06 08:30:23          0 tst.txt
```

asw s3 sync

Instead of uploading individual files, you can synchronize entire directory trees by using `aws s3 sync`:

```
root# aws s3 sync a_local_directory/ s3://linux.book.test
```

When you run the command for the first time, all files of the local directory will be uploaded. If you repeat the command later, only the changes will be made, leaving locally deleted files in the S3 bucket by default (unless you use the `--delete` option).

Other aws-s3 commands

Besides `ls`, `cp`, and `sync`, there are only five other commands: `mv` moves a file, `rm` deletes a file, `mb` creates a bucket, and `rb` deletes the bucket, and finally, you can use `website` to turn a bucket into a static web page. The latter is useful if you want to offer files for public download. A detailed description of the `aws-s3` commands with all their options can be found at *https://docs.aws.amazon.com/cli/latest/reference/s3/index.html*.

Trouble with Cron

On some Linux distributions, `/usr/local/bin` is not included in the `PATH` variable relevant to Cron jobs. In that case, calling `aws` in Cron jobs will fail. A solution to this problem is as follows: Create a link from `/usr/bin/aws` to `/usr/local/bin/aws`, or always specify the full path of the `aws` command in the script.

28.10.3 Encryption and Example

Following Edward Snowden's revelations, you unfortunately have to assume that the NSA and other intelligence agencies have unrestricted access to your data in the cloud—whether the cloud operators like it or not. The fact that your data is transmitted encrypted during upload is pleasing, but it doesn't change this problem. If you want to ensure that your files can only be read by you, you must encrypt them *prior to* uploading.

I often do this by first creating a local backup directory on a server. There I run the backups that seem appropriate to me, whereby I already encrypt all backup files on this occasion. In a second step, I then synchronize the local backup directory to an S3 bucket, to an FTP server of the hosting provider, and so on.

Encrypting and decrypting

In the following example, I assume that the `/local-backup` local backup directory exists. It's advisable to set it up in a separate logical volume with its own file system. All backup scripts and keys are located in `/etc/myscripts`. A passphrase from a file is used for encryption and decryption.

The binary `/etc/key` file with a length of 32 bytes serves as the key. (The 32 bytes size seems small, but it's 256 bits. For symmetrical methods, 128 bits is already considered sufficiently secure.) You can create a random key using the `openssl rand` command—for example:

```
root# openssl rand  32 > /etc/myscripts/key
root# chmod 400 /etc/myscripts/key
```

For encryption, I use the `gpg` command, which is packaged in the `mycrypt` and `myuncrypt` scripts. For simplicity, the encryption is symmetrical:

```
#!/bin/sh -e
# file /etc/myscript/mycrypt
# Use: mycrypt < plain > crypted
gpg -c --batch --cipher-algo AES256 --compress-algo none  \
  --passphrase-file /etc/myscripts/key --passphrase-repeat 0

#!/bin/sh -e
# file /etc/myscript/myuncrypt
# Use: myuncrypt < crypted > plain
gpg -d --batch --no-tty -q --cipher-algo AES256 --compress-
algo none  \
  --passphrase-file /etc/myscripts/key --passphrase-repeat 0
```

The Fine Art of Encryption

The encryption concept presented here is only sufficient for simple cases. If you have higher security requirements or want more flexibility in accessing the keys, you need to consult appropriate technical literature and perform encryption and decryption using asymmetric methods (public key for encryption, private key for decryption).

Local backup

The `backup-cms` Cron script that runs daily at 2:00 a.m. creates a backup of the MySQL or MariaDB `cms` database, compresses and encrypts it, and saves it as `cms-nn.sql.gz.crypt` in `/local-backup/mysql`. The `nn` in the filename represents the day in the month (1 to 31). This means that there are always individual daily backups that go back one month:

```
#!/bin/sh -e
# file /etc/myscripts/backup-cms
db=cms
opts='--skip-opt --single-transaction --create-options --quick  \
      --extended-insert --disable-keys --add-drop-table'
day=$(date "+%d")
fname="/local-backup/$db-$day.sql.gz.crypt"
mysqldump --defaults-file=/root/.my.cnf $mysqlopt $db  \
  | gzip -c  \
  | /etc/myscripts/mycrypt > $fname
```

S3 sync

Another Cron script, which runs daily at 3:00, now only has to synchronize the backup files in `/local-backup` into an S3 bucket:

```
#!/bin/sh -e
aws s3 sync /local-backup/ s3://my.company.backups
```

Test

Now, of course, you need to test the procedure: Do the automatic backups work? Are you also able to download the backup file from the S3 storage on another server, decrypt it, decompress it, and then upload it to a MySQL server? Of course, it's important that you have secure copies of the S3 credentials, as well as the `/etc/myscripts/key` file!

Chapter 29
Firewalls

The title for this chapter is a bit striking—simply because everyone associates something different with the term *firewall*. In fact, this chapter is not only about filters for network packets, but also about network basics and about other techniques for securing them. You will first get to know some fundamental tools to analyze the current status of the network—for instance, to find open ports. A section on the TCP wrapper library shows you how to restrict access to some network services using simple rules. I will then introduce you to firewall configuration tools and the `nft` command of the new *nftables* firewall system.

29.1 Network Fundamentals and Analysis

Before you can secure your computer, you need to get an idea of how network services work, which services are currently running, which ports are open, and the like. This section therefore deals with the fundamentals of TCP/IP and describes some programs to analyze the current network status and, for example, to list all currently active network connections. Before that, Table 29.1 provides an overview of the most important abbreviations.

Abbreviation	Meaning
DNS	Domain Name Service
HTTP	Hypertext Transfer Protocol
ICMP	Internet Control Message Protocol
IP	Internet Protocol
NFS	Network File System
TCP	Transmission Control Protocol
UDP	User Datagram Protocol

Table 29.1 Network Glossary

Internet Protocol

Virtually all common network services are based on IP packets. For example, if an internet user wants to access your computer/server via HTTP or HTTPS, they start a web browser on their computer for this purpose. This

program sends very special IP packets to your computer. If an HTTP server is installed on your computer, it receives the IP packets and responds to the request by sending IP packets back to the client itself.

In addition to the actual data, IP packets contain, among other things, four essential pieces of information: the sender IP address, the sender port, the destination address, and the destination port. This data indicates where the package comes from and where it should go.

IP addresses and ports

The meaning of the IP address should be clear (see Chapter 15). IP ports are used to identify different services. For example, port 443 is usually used to request a WWW document, or port 80 if the access is unencrypted (HTTP instead of HTTPS). Port numbers are 16-bit numbers. Ports up to 1024 are considered privileged and are reserved for server services. The remaining ports can be freely used for your own purposes; however, there are also various numbers that should not be used here because they are often already reserved for specific purposes.

Alias names are defined for many IP port numbers in `/etc/services`. Table 29.2 lists the most important port numbers with the usually valid names and a short explanation.

Name	Port	Function
`ftp`	20, 21	FTP
`ssh`	22	SSH
`smtp`	25	Email
`domain`	53	DNS
`bootps` and `bootpc`	67, 68	DHCP
`http`	80	Web
`pop3`	110	Email
`portmap`	111	Portmap (for NFS)
`ntp`	123	Time (Network Time Protocol)
`netbios-ns`	137	Microsoft/NetBIOS Name Service
`netbios-dgm`	138	Microsoft/NetBIOS Datagram Service
`netbios-ssn`	139	Microsoft File Sharing (SMB, Samba)
`imap`	143	Email
`ldap`	389	LDAP

Table 29.2 Important IP Ports

Name	Port	Function
–	427	Apple Filing Protocol (AFP)
https	443	Web (encrypted)
microsoft-ds	445	CIFS file system (SMB, Samba)
ssmtp	465	Email encrypted (deprecated)
printer	515	Printing with LPD/LPR
–	548	Apple Filing Protocol (AFP)
–	587	Email encrypted
ipp	631	Printing with IPP/CUPS
rmi	1099	Remote Method Invocation (Java)
pptp	1723	PPTP/VPN
nfs	2049	NFS
–	3128	Squid (web proxy)
mysql	3306	MySQL or MariaDB database server
–	5353	Network configuration via Zeroconf/ Bonjour
–	5999–6003	X-Display
–	9100	HP JetDirect Network Printer

Table 29.2 Important IP Ports (Cont.)

IP protocols

There are different protocols for IP packets; most internet services use TCP. This protocol requires an acknowledgement of receipt. However, there are protocols that do not expect such a confirmation, like ICMP (used by `ping`, for example) and UDP (used by DNS and NFS, for example).

IP packet filters

IP packets can be generated by local programs or come into the computer from the outside—that is, via network interfaces. The kernel must then decide what to do with the packages. It can discard the packets or redirect them to running programs or other interfaces. All the packet features described previously can serve as possible decision criteria. To implement a packet filter, you therefore need a way to give the kernel rules on how it should handle certain IP packets. The `nft` command is used for this purpose, and I will introduce it in more detail in Section 29.5.

Determining active network ports

The functional principle of most network services is that they monitor a specific port. If IP packets arrive for this port, the service takes care of processing and replying to them. Packets addressed to unmonitored ports are simply ignored and thus do not pose a threat. To assess the risk to a computer, it is therefore useful to determine a list of monitored ports. Conversely, the first thing an attacker will do is try to find out which ports are active.

netstat

To determine what network activity is taking place on the local computer, the `netstat` command from the `net-tools` package is pretty useful. Depending on the options with which it is called, it provides a wealth of different information. `netstat` can be run by ordinary users, but some options require `root` privileges. I will limit myself here to an exemplary presentation of the command. A reference of the numerous options is given by `man netstat` if needed.

An easy to remember combination of options is `-tulpen`: `netstat` then considers TCP and UDP connections (`-t` and `-u`), displays active sockets (`-l` for *listening*) and the process number of the program (`-p`), enriches the result with various additional columns (`-e`, *extended*), and displays IP addresses in numerical form (`-n`). The following result, shortened for space reasons, was created on a server running Apache, Postfix, Dovecot, and a MySQL server. The result shows at which ports programs are waiting for requests (`LISTEN` status):

```
root# netstat -tulpen
Active Internet connections (only servers)
Proto Recv Send Local Address    Foreign Addr. State
tcp      0    0 127.0.0.1:10023  0.0.0.0:*     LISTEN
tcp      0    0 127.0.0.1:3306   0.0.0.0:*     LISTEN
tcp      0    0 0.0.0.0:587      0.0.0.0:*     LISTEN
tcp      0    0 127.0.0.1:783    0.0.0.0:*     LISTEN
tcp      0    0 0.0.0.0:143      0.0.0.0:*     LISTEN
tcp      0    0 0.0.0.0:22       0.0.0.0:*     LISTEN
tcp      0    0 0.0.0.0:25       0.0.0.0:*     LISTEN
tcp      0    0 127.0.0.1:12345  0.0.0.0:*     LISTEN
tcp      0    0 0.0.0.0:993      0.0.0.0:*     LISTEN
tcp6     0    0 :::587           :::*          LISTEN
...
tcp6     0    0 :::993           :::*          LISTEN
```

If you are only interested in currently active TCP and UDP connections, you can omit the `-l` option. I replaced the local address with `n.n.n.n` in the following listing, and I removed some columns to save space:

```
root# netstat -tuep
Active Internet connections (w/o servers)
Proto  Local Addr.   Foreign Address         State
tcp    n.n.n.n:143   194.118.35.145:53013    ESTABLISHED
tcp    n.n.n.n:22    61.216.145.154:36084    ESTABLISHED
tcp    n.n.n.n:22    61.216.145.154:9224     FIN_WAIT1
tcp    n.n.n.n:22    194.118.35.145:53355    ESTABLISHED
tcp    n.n.n.n:143   194.118.35.145:53003    ESTABLISHED
tcp    n.n.n.n:143   194.118.35.145:52989    ESTABLISHED
tcp    n.n.n.n:143   194.118.35.145:52991    ESTABLISHED
tcp    n.n.n.n:143   194.118.35.145:52988    ESTABLISHED
tcp6   n.n.n.n:443   95.143.172.218:56266    TIME_WAIT
```

lsof

If you want to determine which programs use TCP or UDP ports, the `lsof` command will also help. Using the `lsof -i [protocol]@[hostname][:port]` format, it returns a list of processes that use the specified network resources. The following two commands show all processes that use the UDP protocol or port 22:

```
root# lsof -i udp
ntpd      3696     ntp   16u  IPv4   9026       UDP *:ntp
ntpd      3696     ntp   17u  IPv6   9028       UDP *:ntp
ntpd      3696     ntp   18u  IPv6   9031       UDP ip6-localhost:ntp

portmap   4745  daemon    3u  IPv4  12931       UDP *:sunrpc
rpc.statd 4764   statd    5u  IPv4  12962       UDP *:700
rpc.statd 4764   statd    7u  IPv4  12970       UDP *:39146
...
root# lsof -i :22
COMMAND  PID USER FD TYPE DEVICE SIZE NODE NAME
sshd    5559 root 3u IPv6  14097       TCP *:ssh (LISTEN)
sshd    7729 root 3r IPv6  33146       TCP mars.sol:ssh->mercury.sol:
                                           45368 (ESTABLISHED)
```

nmap

`netstat` and `lsof` can only be run on the local computer and are usually not available to an attacker, who would instead use port scanners. Such programs send packets to the most important ports of a computer and find out which ports are active based on the response. The `nmap` command presented here is the best-known, but by no means the only such program. Most distributions require it to be installed before it can be used for the first time.

The `-sV` option is used by `nmap` to find out which program in which version is responsible for the active ports. The `-O` (the letter O, not the number zero)

option causes nmap to also try to detect the running operating system. The following lines show the results for a web and mail server. (The output is heavily shortened due to space limitations.) Note that determining all this information takes quite some time—in this case, about four minutes:

```
root# nmap -sV -O a-hostname
Starting Nmap 7.70 ( https://nmap.org ) at 2019-05-24 10:49 CEST
Host is up (0.025s latency).

Not shown: 991 closed ports
PORT    STATE SERVICE    VERSION
22/tcp  open  ssh        OpenSSH 7.6p1 Ubuntu 4ubuntu0.3 (Ubuntu ...)
25/tcp  open  smtp       Postfix smtpd
80/tcp  open  http       Apache httpd 2.4.29
143/tcp open  imap       Dovecot imapd (Ubuntu)
443/tcp open  ssl/ssl    Apache httpd (SSL-only mode)
554/tcp open  rtsp-proxy RTSP Proxy Reference Implementation
555/tcp open  rtsp-proxy RTSP Proxy Reference Implementation
587/tcp open  smtp       Postfix smtpd
993/tcp open  ssl/imap   Dovecot imapd (Ubuntu)
Device type: general purpose
Running: Linux 3.X|4.X
OS CPE: cpe:/o:linux:linux_kernel:3 cpe:/o:linux:linux_kernel:4
OS details: Linux 3.10 - 4.11
Network Distance: 8 hops
...

Nmap done: 1 IP address (1 host up) scanned in 237.54 seconds
```

The preceding example cannot even remotely reflect the capabilities of nmap. A good overview of the countless nmap options is provided by the approximately 20-page man page. And if that's not enough for you, the *Nmap Network Scanning* book (published by nmap developer Gordon Lyon in 2009) describes all conceivable basics and details of network scanning in almost 500 pages. You can even read about half of this book for free on the nmap website at *https://nmap.org/book*.

In a collaboration with 10 other authors, I provide an introduction to the use of various other hacking tools and tips on how to defend against attacks in the *Hacking and Security* book (2023), also published by Rheinwerk Publishing.

Never Perform Port Scans for Foreign Servers without Being Asked!

A port scan is considered an intrusion attempt by many administrators. Never use programs like `nmap` to address another computer without being asked! `nmap` is, however, a useful tool for detecting weaknesses in your own network.

29.2 Basic Protection of Network Services

The previous section has shown how you can quickly get an overview of the running network services. The next step is to secure the services as much as possible:

- You should uninstall all network services (programs like Apache, Samba, etc.) that you do not need. Uninstalled programs do not run and therefore cannot pose a threat.
- For the required network services, it's often sufficient to restrict their access to certain clients, such as from the local network. It is, for instance, rarely necessary for a print server to offer its services on the internet!

  For Apache, Samba, MySQL, and numerous other "big" services, the protection must be done in the respective configuration file. Fortunately, there are also many network services that rely on the TCP wrapper library for access control. This enables central configuration (see Section 29.2.1).
- Essential network services should be executed with minimal rights. This is taken care of by your distribution's init system. As far as is possible and reasonable, the init system starts the services without `root` privileges in an account designed for the service or in a `chroot` environment that prevents access to files outside the `chroot` directory.
- As an additional layer of protection, a packet filter firewall is recommended, which uses rules to block packets coming from the internet for various services from the outset (see Section 29.3).
- No program is free of errors. Program errors can enable an attacker to crash the program or even run their own commands by the targeted transmission of manipulated network packets. To minimize the resulting risk, the kernel can monitor the execution of programs by using rules. This approach is called *mandatory access control* (MAC). On Linux, the SELinux and AppArmor methods are widely used for this purpose, which I introduce in Chapter 30.

Updates, logging

The secure configuration of a computer is not a one-time job, but a continuous process. Only regular software updates keep the software on your computer up to date. It is also recommended to take a regular look at the logging files of your computer.

29.2.1 The TCP Wrapper Library

Especially on a LAN server, it rarely makes sense to make all network services globally available. It is sufficient if the services can be used on the local network. A number of network services use the so-called TCP wrapper library for this basic protection. These include in particular the SSH and NFS servers.

/etc/hosts.allow and hosts.deny

The `/etc/hosts.allow` and `/etc/hosts.deny` files control which services may be used from which host. The settings apply only to network services that use the TCP wrapper library or the `tcpd` command for access control. By default, both files are empty; that is, no restrictions apply.

The TCP wrapper library first analyzes `hosts.allow`. If access is explicitly allowed there, the check is done. Otherwise, `hosts.deny` will also be analyzed. If access is denied there, the client receives an error message. *Caution:* In all cases that are not covered by either `allow` or `deny`, rules access will be granted!

You can achieve the most secure configuration possible by generally prohibiting the start of any network service in `/etc/hosts.deny` via `all:all` at first. The `spawn` statement also causes each attempt to start any service to be logged in the `/var/log/deny.log` file. From then on, `/var/log/deny.log` tells you who is trying to use a network service on the computer and when. Not every attempt necessarily constitutes an attack. It is also possible that someone mistyped the host name or IP address:

```
# /etc/hosts.deny
# by default prohibit everything, every connection attempt
# logging
ALL : ALL : spawn (echo Attempt from %h %a to %d at $(date)  \
  >> /var/log/deny.log)
```

In the second step, you can then allow the use of certain services in `/etc/hosts.allow`. The sample configuration shown ahead allows the following IPv4 accesses:

- Access to all services from the local computer (`localhost`)
- SSH access from any computer—that is, also from the internet
- The use of NFS (`portmap` and `mountd`) within the local network

The example assumes that the server is accessible under the names mars and mars.sol, that the local network is operated in the 192.168.0.* address range, and that all computers use the *.sol domain. You can of course use the same pattern to activate other network services, which you initially deactivated globally, for the local network or for any address range:

```
# /etc/hosts.allow
# allow individual services
ALL     : localhost mars mars.sol  : ALLOW
sshd    : 0.0.0.0                  : ALLOW
portmap : 192.168.0.0/24 *.sol     : ALLOW
mountd  : 192.168.0.0/24 *.sol     : ALLOW
```

The syntax within hosts.allow or hosts.deny should be clear from the examples. Each entry consists of three parts separated by colons. The first part specifies the service, the second part specifies the IP address or network name, and the third part specifies the resulting action. You can get a more detailed syntax description via man 5 hosts_access.

IPv6

The TCP wrapper library can also handle IPv6. The IPv6 addresses must be placed in square brackets within hosts.allow and hosts.deny. The following lines allow any IPv6 connection from localhost as well as SSH connections from the local IPv6 network:

```
# /etc/hosts.allow
ALL     : [::1]                       : ALLOW
sshd    : [2001:1234:789a:0471::1/64] : ALLOW
```

Determining TCP wrapper support

You can use ldd to easily determine for yourself whether a particular program uses the TCP wrapper library (libwrap). This often depends on the distribution. The results for cupsd and sshd on Ubuntu look as follows:

```
user$ ldd /usr/sbin/cupsd | grep wrap
user$ ldd /usr/sbin/sshd  | grep wrap
        libwrap.so.0  => /lib/x86_64-linux-gnu/
libwrap.so.0 (0x00007faa9c368000)
```

Thus, cupsd does not use the wrapper library (ldd does not list the library), whereas ssh does.

29.2.2 Starting Network Services without root Privileges

For programs such as Apache or MySQL to do their work, the programs do not need to be run with root privileges. This is why most distributions provide separate accounts for such services, the names of which vary from distribution to distribution. For example, on Ubuntu, Apache runs in the www-

data account and is therefore only allowed to access files that are readable by that account. You can convince yourself of this by using ps axu. By the way, there is *one* instance of Apache that does run with root privileges. However, it is only responsible for starting the other instances and does not perform any other tasks:

```
root# ps axu | grep apache2
root     ... /usr/sbin/apache2 -k start
www-data ... /usr/sbin/apache2 -k start
www-data ... /usr/sbin/apache2 -k start
...
```

Because the init system scripts are basically executed with root privileges, a special mechanism is required to start the network daemon in a different account. In the simplest case, the process is started using the su accountname -c daemon command. However, most network processes provide for more sophisticated mechanisms where the program performs the initialization with root privileges and only then switches to an account with fewer privileges:

- Some programs, such as syslogd, have a separate option to specify the relevant account.
- Apache, MySQL, and some other server services that launch multiple instances also run a control process with root privileges. This process often only fulfills very few tasks, usually the start or termination of instances.

29.2.3 Starting Network Services in a chroot Environment

The chroot rootdir command starts the specified command, using rootdir as the root directory. The command can then only access files located within this directory. To ensure that the program cannot break out of its chroot prison, it must also be run in an account with restricted privileges—not as root.

In real life, however, network services are rarely started using chroot. In fact, many services provide a special option to specify the chroot directory directly. This directory must then contain all the required libraries, configuration files, and so on. If necessary, an init script copies all required files there before starting.

If a network service is monitored by SELinux or AppArmor and the rules are formulated correctly, the use of a chroot environment is unnecessary or provides no additional security. Fedora and Red Hat therefore do not use chroot directories by default and instead rely on the SELinux rules.

29.3 Basic Firewall Principles

The term *firewall* is on everyone's lips, but there is no generally accepted definition for it. Firewall can refer to hardware, when it usually means a computer that is located between the local network and the internet. Most routers also contain basic firewall functions. However, firewall also often refers to a program that is installed on the computer and is intended to improve security if configured correctly. Some distributions include tools for configuring the firewall.

In this book, I use the term *firewall* to refer to the protection of TCP/IP traffic by a packet filter. Such a filter analyzes all network packets entering and leaving the computer. Depending on whether all rules are followed, the packets are allowed to pass or are blocked. Details on how to configure such a packet filter follow in the next section. The first thing to do here is to clarify the terminology.

Firewalls for private computers

The necessity of firewalls on private computers is controversial. For example, the Ubuntu developers believe that there are no network services running on a desktop installation of Ubuntu anyway that need to be protected. In a minimal installation, this is basically correct, but it does not always stop there: as soon as the user installs Samba to make their own files available to other users via a network directory, then that is the first service that can be attacked from the outside.

So long as the computer is operated at home and obtains internet access through a router, the router provides a certain degree of protection from the outside—first through the widely used NAT method (only for IPv4), and second perhaps also through a firewall running on the router.

The situation is completely different if you use your notebook computer to check emails in the unsecured wireless network of a hotel. By then, at the latest, a firewall should also be a matter of course on private PCs.

Firewalls for local networks

In the case of a corporate LAN, the desire for good security is even more pronounced. At the same time, there are also better conditions in terms of infrastructure. In reality, a dedicated computer often takes care of internet access for the company and its security. All other network services run on other computers.

29.3.1 Netfilter and Nftables

From netfilter to ntftables

In the course of Linux history, there have been several packet filter systems in the kernel. The last 15 years or so have been dominated by the netfilter system and the associated `iptables` command. Most recently, however, there has been a shift toward the nftables system. The development of this

packet filtering system started back in 2008. nftables fixes some basic netfilter deficiencies and works much more efficiently. nftables has been used in most distributions since around 2020.

To enable a smooth changeover, nftables can be controlled not only by the new `nft` command, but also by the established `iptables` command. This is important because a large number of common firewall tools and scripts require `iptables`.

In some distributions, you can use `alternatives` in order to switch between the deprecated original `iptables` command (*legacy*) and an implementation of `iptables` based on nftables. If you plan to build your own firewalls manually, you should not choose one or the other, but go for `nft` right away!

29.4 Firewall Configuration Tools

Many distributions help their users to configure the firewall by providing convenient user interfaces. This allows you to set up or modify a simple firewall virtually at the click of a mouse. All the programs described in this section also set up rules for IPv6. A lot of packet filter knowledge is usually hidden behind the fairly simple configuration. So the result is often better protection than a homemade solution.

The disadvantage of predefined firewalls is their black box behavior; if the effect of the filter is documented at all, it is mostly poor. You don't know what the filter protects you against, nor what the side effects of the filter are. It can happen that Linux beginners, after days of searching for why printing on the network is impossible, finally find the firewall to be the culprit. Trying to adjust the packet filter accordingly now will probably fail, especially if basic firewall knowledge is missing. The predictable outcome is that the firewall will simply be turned off!

29.4.1 Debian

Debian does not provide a firewall by default and does not know any distribution-specific configuration tools. However, current Debian versions provide the FirewallD program developed by Red Hat. The Debian Wiki explicitly recommends the use of FirewallD.

After installing `firewalld`, the firewall is active immediately. The network interfaces are assigned the `public` zone, which by default allows only the DHCPv6 client and SSH services. If you want to use Samba or set up server services, you have to define appropriate exception rules by using `firewall-cmd` or `firewall-config` (see Section 29.4.3).

29.4.2 Fedora and RHEL

FirewallD

Fedora and RHEL have been using Red Hat's own FirewallD development for many years. In addition, in recent years, more and more other distributions have also switched to FirewallD, including Debian (optional) and SUSE (by default).

The main advantage of FirewallD over many other systems is that changes are made dynamically at runtime. In contrast, the previous static system required the firewall to be temporarily shut down when changes were made and then rebuilt. Not only is this a security risk, but it also causes existing network connections to be interrupted.

Backend

FirewallD is started by systemd and configured by files in `/etc/firewalld` and `/usr/lib/firewalld`. Current versions use nftables as the backend (see the `FirewallBackend` setting in `/etc/firewalld/firewalld.conf`).

Zones

The `firewalld` background process is responsible for managing the firewall. Configuration tools communicate with the daemon via D-Bus. A central basic idea of FirewallD is that each network interface is assigned a zone. A *zone* in the sense of FirewallD is a collection of rules for a specific usage purpose. The following list very briefly describes some zones:

- `block` blocks all network traffic. The sender receives an ICMP error message.
- `drop` blocks all network traffic. The sender is not informed.
- `Trusted` allows any network traffic. This zone is intended for well-secured local networks, but not for Wi-Fi connections.
- `external` blocks most ports and enables masquerading (IPv4). The zone is intended for the interface that connects to the internet on a router.
- `home` and `internal` block most ports, but accept Samba (client only), CUPS, and Zeroconf/Avahi/mdns. The zones are intended for computers on a local network that is considered reasonably secure. If you want to use this zone and share Windows network directories yourself, you must also enable the **Samba** service.
- `public` is similar to `home`, but it also blocks CUPS and Samba client functions. The zone is intended for internet use in insecure networks—for example, in a public wireless network.
- `FedoraWorkstation` and `FedoraServer` are used by default for network interfaces of Fedora installations. `FedoraServer` blocks all services except SSH and the DHCP client configuration. `FedoraWorkstation` is much more tolerant and also accepts Samba client connections and traffic through all ports between 1025 and 65535.

The zone rules can be changed with the exception of `block`, `drop`, and `trusted`. If necessary, you can also define your own zones.

Zone assignment in ifcfg-xxx

By default, RHEL assigns all interfaces to the `public` zone. In current Fedora versions, the `FedoraWorkstation` or `FedoraServer` zone is used instead. If an interface is supposed to use a different zone, it's specified in the configuration file with the `zone` keyword:

```
# file /etc/NetworkManager/system-connections/
<interfacename>.nmconnection
[connection]
...
zone=trusted
```

The `firewall-config` user interface is available for configuring the firewall (see Figure 29.1). The package of the same name must be installed before the first start. Unfortunately, its operation is quite confusing. After it starts, you will see the so-called runtime configuration of the various zones. Changes made here apply immediately to the relevant zone, but they are not saved and are thus lost when the computer is restarted.

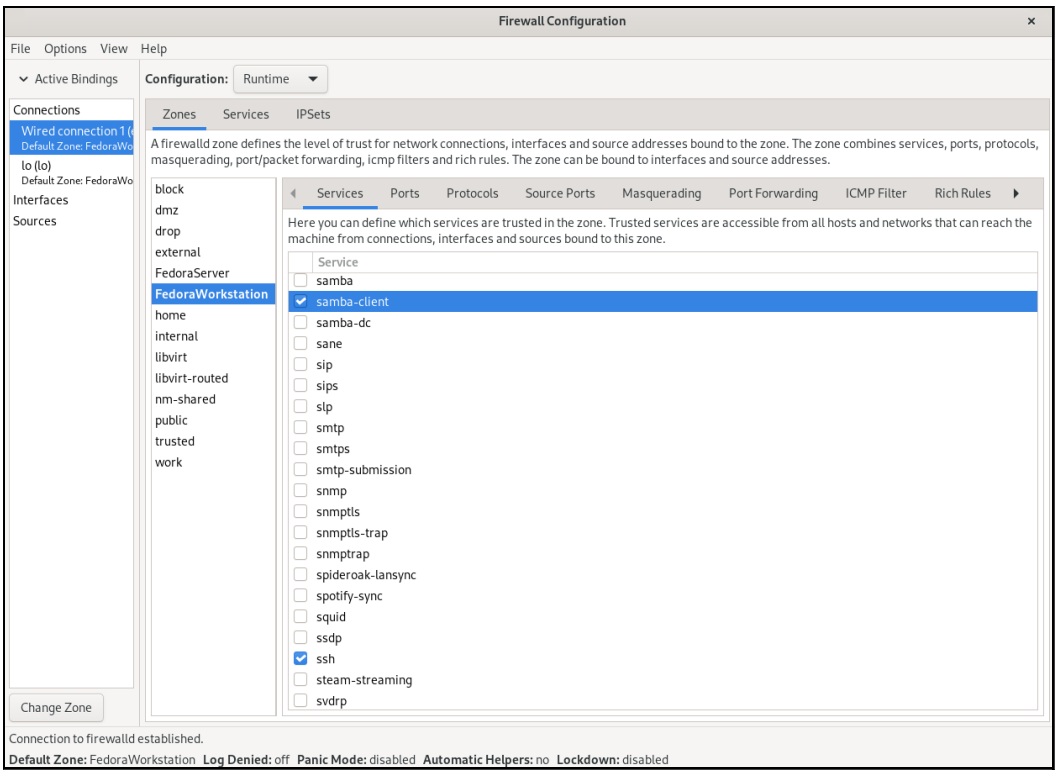


Figure 29.1 Graphical User Interface for FirewallD Configuration

User interface

Using **Options • Change Default Zone**, you can set which zone applies by default—that is, for all network interfaces where another zone is not explicitly set. To change the firewall zone individually for a particular interface, you need to run **Options • Change Zones of Connections**.

To permanently change the definition of a zone, you can switch to the **Configuration = Permanent** view in the configuration program. Annoyingly, changes you previously tried in the **Runtime** view now have to be repeated.

Disabling the Firewall

The configuration program does not provide a direct option to disable the firewall altogether. If you want this, you need to run **Options • Change Default Zone** and set the `trusted` zone. By default, all interfaces are assigned to the default zone. Due to `trusted`, all network traffic is allowed.

Note, however, that this change to the default zone does not affect interfaces that you have explicitly assigned to a different zone. If you have such interfaces, you have to edit them separately via **Options • Change Zones of Connections**.

If you want to set up a completely separate firewall, you must uninstall the `firewalld` package.

29.4.3 firewall-cmd

In the terminal, you can make FirewallD configuration changes via `firewall-cmd`. By default, the changes are only made dynamically, but not saved. If you want the changes to be permanent, you must also specify the `--permanent` option and then explicitly activate the changes via `firewall-cmd --reload`. Custom rules are stored in the `/etc/firewalld` directory.

Status

To get an overview of the current state of the firewall, you must execute the following two commands:

```
root# firewall-cmd --state
running

root# firewall-cmd --get-active-zones
internal
  interfaces: enp0s8
external
  interfaces: enp0s3
```

You now know that the firewall is active and which network interfaces are assigned to which firewall zones.

Zones The following commands explore which zones exist and which of them are active:

```
root# firewall-cmd --get-zones                          (list all zones)
  FedoraServer FedoraWorkstation block dmz drop
  external home internal public trusted work

root# firewall-cmd --get-default-zone                   (determine default zone)
  FedoraWorkstation

root# firewall-cmd --get-zone-of-interface=enp4s0       (zone for one interface)
  FedoraServer
```

`firewall-cmd --list-all` provides an overview of all rules that apply to the default zone or to the zone selected with `--zone`. The following listing shows that the firewall allows the DHCPv6 client, SSH, Samba, and Samba client services and communication for all ports greater than 1024. Otherwise, incoming server requests will be blocked. Outgoing connections from the client (HTTP, FTP, etc.) are allowed:

```
root# firewall-cmd --zone=FedoraWorkstation --list-all
FedoraWorkstation (active)
  target: default
  icmp-block-inversion: no
  interfaces: enp1s0
  sources:
  services: dhcpv6-client samba samba-client ssh
  ports: 1025-65535/udp 1025-65535/tcp
  protocols:
  forward: no
  masquerade: no
  forward-ports:
  source-ports:
  icmp-blocks:
  rich rules:
```

You can change the default zone using `firewall-cmd --set-default-zone`. You can change the assignment of a zone to an interface by using `--change-interface`. The following command ensures that `enp4s0` will be assigned to the `FedoraWorkstation` zone in the future, no matter which zone the interface previously belonged to:

```
root# firewall-cmd --permanent --zone=FedoraWorkstation \
                   --change-interface=enp4s0
```

Note, however, that zone mapping can also be done through /etc/sysconfig/network-scripts/ifcfg-xxx. The ifcfg files have priority over changing the zones!

Services

To list all services known for FirewallD, you can use the following command:

```
user$ firewall-cmd --get-services
  RH-Satellite-6 amanda-client amanda-k5-client ...
  vnc-server wbem-https xmpp-bosh xmpp-client xmpp-local xmpp-server
```

To enable a service or an IP port for a zone, you must run the following commands:

```
root# firewall-cmd --permanent --zone=FedoraWorkstation \
  --add-service=<name>
root# firewall-cmd --permanent --zone=public --add-port=<nnn>/tcp
root# firewall-cmd --permanent --zone=public --add-port=<nnn>/udp
root# firewall-cmd --reload     (activate changes)
```

In this context, <name> is the name of a service (such as https or nfs), while <nnn> is the number of a port.

Documentation

Comprehensive documentation for FirewallD as well as a lot of usage examples for firewall-cmd can be found at *https://fedoraproject.org/wiki/FirewallD*.

29.4.4 SUSE

SUSE has been using the FirewallD system developed by Red Hat since version 15. Instead of firewall-config, a YaST module is provided for configuration. However, this module is completely immature and provides only a few functions. In my tests on openSUSE 15.5, it didn't even show the usual Ethernet interface. For firewall configuration, it's better to use the firewall-cmd command described in the previous section.

29.4.5 Ubuntu

ufw

Ubuntu does not set up a firewall by default, and there are no graphical configuration tools. For this purpose, Ubuntu contains the ufw command (for *uncomplicated firewall*). It allows the definition of firewall rules in a relatively simple syntax. Internally, ufw consists of Python code that uses iptables as a backend. However, ufw has found little acceptance outside of Ubuntu. If you already have experience with packet filters, you will quickly reach your goal when using ufw: ufw enable activates the firewall. The firewall is activated immediately and also in the future at every computer start.

`ufw disable` deactivates the firewall again, while `ufw default allow` or `ufw default deny` specifies whether incoming packets are generally accepted or rejected. By default, `deny` is used, which means that with the activation of the firewall, no network service running on the computer can be accessed from the outside anymore!

In addition, you can use `ufw allow/deny nnn` and `ufw allow/deny service` to define rules that apply to specific IP ports and protocols, respectively. `ufw status` provides information about the current state of the firewall. By default, `ufw` takes care of both IPv4 and IPv6. If you want to block IPv6 traffic altogether, you can use the `IPV6=no` setting in `/etc/sysconfig/ufw`:

```
user$ sudo -s
root# ufw enable
root# ufw allow ssh
root# ufw status
Status: active
To     Action    From
-      ------    ----
22     ALLOW     Anywhere
22     ALLOW     Anywhere (v6)
```

If required, you can formulate firewall rules for specific interfaces or IP address ranges. Details are explained in `man ufw`.

Custom rules are stored in `/lib/ufw/user.rules` and `user6.rules` (for IPv6). In addition, `ufw` takes into account the rules files from the `/etc/ufw/` directory.

Profiles Furthermore, when installing network services, suitable rule files are saved in the `/etc/ufw/applications.d` directory. `ufw app list` provides the list of profiles defined in this way. Using `ufw allow/deny`, you can enable such profiles (i.e., enable the corresponding ports in the firewall) or block them:

```
root# ufw app list
Available applications:
  Apache
  Apache Full
  Apache Secure
  CUPS
  Dovecot IMAP
  Dovecot POP3
  Dovecot Secure IMAP
  Dovecot Secure POP3
  OpenSSH
  Postfix
  ...
```

```
root# ufw app info "Apache Full"
  Profile: Apache Full
  Title: Web Server (HTTP,HTTPS)
  Description: Apache v2 is the next generation of the omnipresent
  Apache web server.
  Ports: 80,443/tcp
root# ufw allow "Apache Full"
```

Note that application profiles are installed by default, but not activated! The profiles are only meant to give you some guidance if you want to set up a firewall.

Documentation

For more information and examples for `ufw`, refer to `man ufw` and to the following pages:

- *https://ubuntu.com/server/docs/security-firewall*
- *https://wiki.ubuntu.com/UncomplicatedFirewall*

29.5 Custom Firewall Built Using nft

This section contains two small examples to show that creating simple firewalls is by no means witchcraft. The examples are intended for anyone who wants to better understand and experiment with firewalls.

> **Warning**
>
> I don't want to give the impression with this section that you can replace professional firewall tools with your own 20-line or 30-line scripts! On the contrary, I advise that you use the firewall tools of your distribution or use distribution-independent firewall configuration tools.

29.5.1 Nftables: Basic Principles

Basically, a packet filter firewall works as follows: Network packets reach the kernel via interfaces. The kernel decides what should happen to the packets on the basis of specific rules. The two main actions are to either redirect the packet to another interface or block it.

Based on this minimal concept, there are of course countless variants. In some cases, it's necessary to modify packets (e.g., for NAT). You may want to count packets that match certain criteria, log actions (*logging*), redirect packets to different interfaces depending on the load (load balancing), and so on. However, all this is far beyond the scope of this introduction.

Tables with chains of rules

Separate terminology is used for nftables. Firewall *rules* are always part of a *chain*. *Tables*, in turn, consist of multiple chains. Unlike `iptables`, there are neither predefined tables nor predefined chains. Thus, you can compose as many custom tables as you like from chains also defined by yourself.

Protocol families

Tables, chains, and rules apply to different types of network packets associated with the following protocol families:

- `ip` (IPv4 only)
- `ip6` (IPv6 only)
- `inet` (rules that apply equally to IPv4 and IPv6)
- `arp` (level 2 rules; are analyzed before level 3 rules)
- `bridge` (switching rules)
- `netdev` (low-level rules; are analyzed before all other rules and enable, for example, a particularly efficient defense against DDOS attacks)

Hooks for address families

Unlike Netfilter and other firewall systems, nftables does not have predefined tables or rule chains for specific tasks. Rather, you can connect chains to different processing phases predefined in the kernel via so-called hooks. The nftables documentation speaks of address families in this context.

The following hooks are available for the `ip`, `ip6`, and `inet` protocol families (see Figure 29.2):

- **`ingress`**
  At this point, it isn't even clear which protocol and which network layer the packet is assigned to. In this respect, only very general rules can be placed here.
- **`prerouting`**
  In routing, the kernel decides where to route a packet. Packets can be filtered beforehand via the prerouting hook.
- **`input`**
  Packets coming from the outside to be processed first pass through the rules of tables associated with the input hook.
- **`forward`**
  At this point, rules can be anchored that apply to packets which are supposed to be forwarded to another computer but not processed locally.
- **`output`**
  The output hook is responsible for packets that are to be generated by local processes (e.g., a web browser, a mail server, or an SSH client) and then routed to the outside.

- **postrouting**
  The postrouting hook allows postprocessing of packages before they leave the kernel.

If you have worked with `iptables` previously, the protocol families will look familiar. In fact, netfilter and nftables use the same kernel functions to anchor rules. Only the way the rules are formulated and managed has changed with the move from netfilter to nftables.

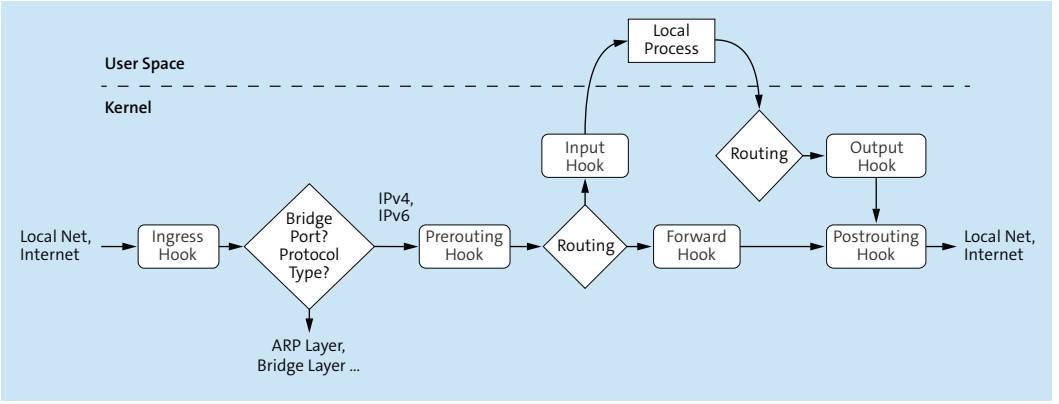


Figure 29.2 Simplified Representation of the Nftables System for IPv4 and IPv6 Packets

"nft" replaces "iptables", "ebtables", and "arptables"

Figure 29.2 is a simplified representation of nftables in that it only considers a subset of the functions. Not only is nftables responsible for IPv4 and IPv6 protocols, but also for the Address Resolution Protocol (ARP) and Ethernet frames. In the Netfilter past, these tasks were distributed across the `iptables`, `arptables`, and `ebtables` commands. nftables has merged these functions into one framework.

However, I only consider the processing of IP packets in this book. A more complete presentation of how the kernel works, as well as nftables, can be found at *https://wiki.nftables.org/wiki-nftables/index.php/Netfilter_hooks*.

Determining the firewall status

`nft list tables` lists all tables currently known by nftables. `nft list table <family> <name>` shows which chains and rules the given table is composed of. The following output is from a Fedora system with FirewallD active:

```
root# nft list tables
table inet firewalld
table ip firewalld
table ip6 firewalld
```

```
root# nft list table inet firewalld

table inet firewalld {
        ct helper helper-netbios-ns-udp {
                type "netbios-ns" protocol udp
                l3proto ip
        }

        ct helper helper-tftp-udp {
                type "tftp" protocol udp
                l3proto inet
        }

        chain raw_PREROUTING {
                type filter hook prerouting priority raw + 10;
                  policy accept;
                icmpv6 type { nd-router-advert, nd-neighbor-solicit }
                  accept
                meta nfproto ipv6 fib saddr . iif oif missing drop
        }
    ...
```

`nft list ruleset` creates a listing of all tables, chains, and rules. The resulting code has the same syntax as the previous listing. If you redirect the output to a file, you can restore the firewall without any changes at a later time:

```
root# nft list ruleset > myrules
...
root# nft -f myrules
```

If no firewall is active, then `nft list ruleset` returns no output at all. However, there will be no error messages displayed.

Reset

If you want to set the firewall to an unsecured default state, you need to run `nft flush ruleset`. This will delete all tables, chains, and rules. The kernel accepts any correct IP packet; there is no firewall protection at all. You can use `nft flush ruleset <family>` to restrict the flush command to a specific protocol family.

29.5.2 Defining Rules

Calling nft as a command

There are three ways to define your own rules using the `nft` command. The "classic" approach is to call the command individually for each change to a table, chain, or rule. The syntax of the file then looks like the following example:

```
root# nft add table inet mytable
root# nft add chain inet mytable mychain \
          '{ type filter hook input priority 0; }'
root# nft add rule  inet mytable mychain tcp dport 22 accept
...
```

The first command creates the new table `mytable`, the second creates the new chain `mychain` in it. This chain is connected to the input hook as a filter. (Other permitted `type` specifications are `nat` and `route`.) As long as there is only one chain, `priority` does not matter. However, if there are multiple chains for a specific task (e.g., for the `ip` protocol type and the `input` protocol family), they are applied in the order of their priority. Rules with a low priority value have higher priority.

The third command adds the first rule to `mychain`: TCP packets for IPv4 and IPv6 with *destination port* 22 are supposed to be accepted. Provided that an SSH server is running on the computer, this rule allows the creation of an SSH connection from the outside.

nft rule files

Instead of compiling the firewall in a cumbersome way using countless individual commands, it is a good idea to simply write the firewall code in a text file. This file is structured by bracketed levels of tables, chains, and rules, which eliminates a lot of redundancy and makes the code much more readable. The three commands presented previously can be represented as follows:

```
# Rules file myfirewall.txt
table inet mytable {
        chain mychain {
                type filter hook input priority 0;
                tcp dport 22 accept
                ...
        }
}
```

This rule file can then be read via `nft -f`:

```
root# nft -f myfirewall.txt
```

nft scripts

But it can get even more elegant! You can formulate the firewall code as a script. To do this, you want to begin the script file with `#!/sbin/nft -f` and make the file executable via `chmod +x`:

```
#!/sbin/nft -f
# nft script file my-firewall-script.nft
table inet mytable {
        chain mychain {
```

```
                type filter hook input priority 0;
                tcp dport 22 accept
                ...
        }
}
```

Then you can execute the file like a script:

```
root# ./my-firewall-script.nft
```

Activating the firewall automatically

Along with the `nftables` package, the systemd service of the same name is installed. If you enable this service using `systemctl enable --now`, it reads the `/etc/sysconfig/nftables.conf` (Fedora/RHEL) or `/etc/nftables.conf` (Debian/Ubuntu) file when the computer boots up. There you can either place your own firewall code directly or use `include` statements to read other files.

29.5.3 Syntax for Firewall Rules

In the following sections, I assume that you formulate your rules in a file. Here I focus on filter rules:

```
table ip|ip6|inet <tablename> {
        chain <chainname> {
                type filter hook input|output|... priority <n>;
                # Filter rules
        }
}
```

In the following listings, I only provide the rules and omit the two outer bracket levels. Also note that the following set of rules by no means represents the full range of nftables. Instead, I will focus here on the most important rule types. Countless other variants are summarized on the following page—but the reading time needed is much more than the promised 10 minutes: *https://wiki.nftables.org/wiki-nftables/index.php/Quick_reference-nftables_in_10_minutes*.

Comments and multiline statements

Comments are introduced with the `#` character as usual. Statements may extend over several lines if it is clear to `nft` that the statement is still incomplete (e.g., because of an open parenthesis) or if you end the line with `\`:

```
# OK
icmpv6 type {echo-request,
             nd-neighbor-advert} accept
# also OK
icmpv6 type  {echo-request, nd-neighbor-advert} accept
```

```
# not OK
icmpv6 type
  {echo-request, nd-neighbor-advert} accept
```

Properties of the chain

The first statements of a chain specify the basic properties—that is, the type of chain (`filter`, `nat` or `route`), the connection (hook) to an address family, the priority, and the default behavior (usually `policy accept`, alternatively `policy drop`):

```
type filter hook input priority 0; policy drop;
```

It's common to separate the properties with semicolons. `nft` accepts the statements also without semicolons, but then you have to specify the default `policy` in a separate line:

```
type filter hook output priority 0
policy accept
```

Criterion and action

Most rules consist of a criterion (*match*) and an action (*statement*) that is executed when the criterion is met. The two most important actions are `accept` and `drop`. In addition, there are numerous other actions, such as `reject` (the sender is informed about the error), `counter`, `log` (logging), or `limit` (limit the number of packets per time unit). It's possible to formulate multiple criteria and multiple actions in one rule at the same time:

```
ip protocol igmp limit rate 4/second accept
```

In this case, the condition is `ip protocol igmp`, which means it must be a packet of the Internet Group Management Protocol (IGMP). Such packets are accepted, but only a maximum of four per second.

accept or drop

Many rules end with the keyword `accept` or `drop`. The rule thus determines what should happen to the packet, provided that certain conditions are met. By default, `accept` is the default behavior for filter rules. So if a rule chain does not end with the `drop` action, the packet passes the filter. If you want `drop` to be the default behavior, either opt for the `policy drop` property or specify `drop` as the last rule of a chain without any preceding conditions.

In general, the processing of a rule chain ends with `accept` or `drop`. For this reason, if one of the two actions occurs due to an applicable criterion, the other rules are no longer considered for this packet.

Port test (TCP/UDP)

`tcp sport` (source port) tests from which port a packet was sent. `tcp dport` (destination port) analyzes to which port the packet is addressed. Similarly, these conditions can also be formulated for the UDP protocol. Instead of port numbers, you can also specify number ranges (`1-1023`), names (`dport https`), or listings formulated in curly brackets:

```
tcp dport 22
tcp sport {80, 443}
udp dport ssh udp sport ssh
```

Address test (IPv4/IPv6)

`ip saddr` and `ip daddr` check the source and destination addresses of a packet, respectively. Address ranges and negations are allowed as well:

```
ip saddr 192.168.172.1
ip daddr 10.0.0.0/24
ip daddr != 1.2.3.4
```

`ip6 saddr` and `ip6 daddr` perform similar tests for IPv6 addresses or address ranges:

```
ip6 saddr 1234:5678:9abc:def0:1234:5678:9abc:def0
ip6 daddr ::/64
```

Status

`ct state <name>` checks whether a packet can be assigned to an existing connection. `ct` here stands for *connection tracking*. Permitted statuses are `established`, `invalid`, `new`, `related`, and `untracked`:

```
ct state {established, related}
```

ICMP

`ip protocol icmp` captures all ICMP packets. Similarly, `ip6 nexthdr icmpv6` applies to all ICMPv6 packets:

```
ip  protocol icmp    # all ICMP packets
ip6 nexthdr  icmpv6  # all ICMPv6 packets
```

To formulate rules that only apply to certain types of ICMP packets, use `icmp type <name>` or `icmpv6 type <name>`, whereby you can list multiple types in curly brackets as usual:

```
icmp   type  echo-request                       # selected ICMP
icmpv6 type {echo-request, nd-neighbor-advert}  # packets
```

Caution with ICMPv6 Drops

Due to the danger of ICMP flooding attacks, it's usually not a good idea for servers to accept ICMP packets without restrictions. It's best to set more restrictive rules in combination with rate limiting (discussed ahead). Note, however, that unlike IPv4, IPv6 will not work without ICMP. So you cannot simply block ICMPv6.

Interfaces

`iifname <name>` and `oifname <name>` check from which interface (input interface) a packet originates and to which interface it should be redirected,

respectively. Strictly speaking, the syntax is `meta iifname <name>`, but the `meta` keyword is optional in this context:

```
iifname "lo"      # packets from the loopback interface
oifname "virbr0"  # packets to the virtual bridge virbr0
```

Counter

Using the `counter` keyword after the criterion, nftables counts how often the condition is true and how many bytes are affected by the rule. These readings are then included in the output of `nft list`. For example, the following rule accepts all ICMPv6 packets and counts them simultaneously:

```
ip6 nexthdr ipv6-icmp counter accept
```

The following output by `nft list` shows that the firewall has allowed 33 ICMPv6 packets with a size of 3456 bytes to pass through so far:

```
root# nft list ruleset
...
ip6 nexthdr ipv6-icmp counter packets 33 bytes 3456 accept
```

Limit

You can use `limit rate <n>/<period>` to limit the following action (usually `accept`) to a given number of packets within a time unit (`second`, `minute`, `hour`, `day`):

```
# Accept five pings per minute
icmp type echo-request limit rate 5/minute accept
```

Note that exceeding the limit does not automatically cause the packet to be discarded. Instead, the other rules will then be analyzed. For example, if the `ct state {established, related} accept` rule follows later, a longer lasting ping will fall under this rule and the packet will be accepted after all. It is therefore sometimes necessary to explicitly discard corresponding packets by using `limit rate over ...`:

```
# Accept five pings per minute
icmp type echo-request limit rate 5/minute accept
# discard all further pings
icmp type echo-request limit rate over 5/minute drop
```

The same rules can be formulated in a simpler way. Either the ping is accepted by the first rule or the second rule is applied:

```
# Accept five pings per minute
icmp type echo-request counter limit rate 5/minute accept
# discard all further pings
icmp type echo-request  drop
```

Logging

The `log` keyword causes the metadata of the package to be logged by the kernel when the previously formulated conditions are met. The logging messages can be read via `dmesg` and `journalctl`. Be careful not to log too much, though!

You can prefix the logging messages with a character string using `prefix "message"`. This helps to identify the message:

```
# Log packet only
<bedingung> log prefix "mylogprefix"
# discard packet and log
<bedingung> log prefix "mylogprefix" drop
```

29.5.4 Simple Protection of a Web Server

Prerequisites for this chapter

This example is about securing a simple web server with a firewall. I assume here that you will deactivate any distribution-specific firewall beforehand. For many distributions, this is how it works:

```
root# systemctl disable --now firewalld
```

rules

The basic idea of the firewall is that the ports for SSH, HTTP, and HTTPS remain open and that all ICMP packets are accepted. Internal network traffic is permitted, while (almost) everything else is blocked. To ensure that the firewall always starts from scratch in case of multiple tests, `flush ruleset` first deletes all previously valid rules:

```
#!/sbin/nft -f
# Delete all existing rules
flush ruleset

# Table applies to IPv4 and IPv6
table inet mytable {
    chain mychain {
                # Input filter
        type filter hook input priority 0;

        # Accept packets from the loopback interface
        iifname lo accept

        # Accept packets of existing connections
                ct state {established,related} accept

        # accept 200 ICMP packets per minute accept,
                # block everything else
                icmp type echo-request limit rate 200/minute accept
```

```
                icmp type echo-request drop
                ip6 nexthdr icmpv6 limit rate 200/minute accept
                ip6 nexthdr icmpv6 drop

        # Accept packages for the web server
        tcp dport {http, https} accept

        # Accept packages for the SSH server, but not
        # in excess
        tcp dport ssh limit rate 200/minute accept

        # discard all other packages
        drop
    }
}
```

Do not lock yourself out!

If you set up a firewall in a virtual machine or on a root server that you administrate exclusively via SSH, you run the risk of locking yourself out. This happens when you block all SSH packets by means of firewall rules.

A simple precaution can be to formulate your rules in a file, read them through a script, and then deactivate them after 60 seconds:

```
#!/bin/bash
nft -f myfirewall.nft
sleep 60
nft flush ruleset
```

You now have 60 seconds to try out the firewall. (Of course, you can also choose a longer timeout.) Even if your firewall overshoots the target, it will be removed afterward without you having to intervene in any way.

29.5.5 More Examples

If the `nftables` package is installed, you will find some scripts for various applications in the following directory:

/usr/share/doc/nftables/examples

Of course, you can also find a large number of examples on the internet. In particular, you should take a look at the following pages:

- *https://wiki.gentoo.org/wiki/Nftables/Examples*
- *https://wiki.archlinux.org/title/Nftables#Examples*
- *https://wiki.nftables.org/wiki-nftables/index.php/Main_Page*

Chapter 30
SELinux and AppArmor

SELinux and AppArmor are kernel extensions that monitor running processes and ensure that they follow certain rules. SELinux is used by default on Fedora and RHEL, AppArmor on Debian, SUSE, and Ubuntu. In various other distributions, the protection systems can be optionally activated.

This chapter provides an introduction to the operation and configuration of SELinux and AppArmor. It also describes how you can identify rule violations and how to respond to them.

30.1 SELinux

On Linux, the traditional system for managing access rights normally applies: each program runs in a user account. This account determines which (device) files the program is allowed to access. You will certainly remember that in Chapter 9 I described that each file is assigned to a user and a group. The rwx bits for user, group, and others decide which users are allowed to read or modify the files.

Ordinary programs use the account of the user who started the program. Network services, database servers, and so on are started with `root` privileges, but for security reasons they often switch to another account with restricted privileges immediately after startup.

Although this rights system is extremely simple, it only offers limited configuration options. If an attacker manages to take control of a program, he can access countless files that the program normally does not need. It is particularly bad if the attacker gains control over a program with `root` privileges or if they can use workarounds to ensure that their own code is executed with `root` privileges. This allows them to manipulate the computer without any restrictions, install and launch their own programs, and so on.

You may wonder how an attacker can gain control over a program. Almost always, bugs in the program code are exploited. For example, sending manipulated data triggers a so-called buffer overflow. This error is in turn used to inject the program with its own code and execute it. Of course, there are other methods—but it always boils down to abusing security vulnerabilities of the program in order to use or manipulate the program for purposes other than intended.

Security measures

There are no error-free programs, and there probably never will be in the future, apart from tiny trivial programs. For this reason, all possible procedures have been developed over time to minimize the risks caused by program errors. Established security measures include avoiding daemons with `root` privileges as much as possible, installing as few commands with the `setuid` bit as possible (see Chapter 9, Section 9.6.1), and prohibiting the execution of code in the stack using Red Hat's Exec Shield kernel extension.

SELinux

The SELinux kernel extension, originally developed by the NSA as open-source code, goes one step further. The purpose of this extension is for the kernel to monitor the execution of programs based on rules. This approach is called *mandatory access control* (MAC). If a rule is violated, SELinux prevents the operation or logs a warning. The rule-violating program is not terminated by SELinux. However, it depends on the program how it reacts if it cannot access a certain file or use a network interface.

MAC rules allow much tighter security control than the Unix access system. They can be used to generally prohibit a program from accessing certain directories or network functions, independent of Unix access rights or accounts. As these rules are monitored at the kernel level, they still apply even if the program gets out of control due to an error or security flaw.

SELinux Is Clean, Though the Code Was Developed by the NSA

The *National Security Agency* is a US intelligence agency that has attracted a lot of negative press due to its extensive monitoring of all internet traffic. Although SELinux originated in intelligence circles, there is no risk that Linux has been extended to include surveillance functions in this way: the SELinux code is public, has been checked and improved by many independent experts, and is part of the official kernel. If the NSA monitors you and others, it certainly isn't through a backdoor in SELinux.

SELinux rules

Without appropriate rules, SELinux is ineffective. Whether a system is made more secure by SELinux thus depends primarily on the quality of the rules. Among the mainstream distributors, only Red Hat has so far invested a considerable amount of time and effort into developing such rules. In a way, Fedora serves as a testing vehicle. What proves itself there is ultimately incorporated into the Red Hat Enterprise versions (RHEL).

Criticism of SELinux

SELinux is not without controversy. The two most important points of criticism are as follows:

- Files must be tagged with extended attributes (EAs; see Chapter 9, Section 9.7.2) to ensure the interaction with SELinux. This requires EA-compatible file systems and often causes problems with updates and backups.

- The biggest problem with SELinux is its huge complexity. Even securing the most important network services requires thousands of rules. Few experts are in a position to judge the effectiveness of these rules. Average Linux users are not able to adapt SELinux rules to their own requirements.

  SELinux is therefore rightly considered a black box by the majority of its users. The complexity almost inevitably leads to implementation errors and tempts you to switch off the system completely when the first problem crops up.

Alternatives

The most popular alternative to SELinux is the AppArmor system used by SUSE and Ubuntu (see Section 30.2). Besides that, there is another MAC system in the kernel: Smack, primarily used in embedded Linux systems.

30.1.1 Internal Workings of SELinux and Usage

SELinux is based on two pillars: first, it requires the correct identification of all files and processes by means of a so-called security context, and second, it is based on rules that must be adhered to by the monitored processes.

Security context

SELinux requires that every object (e.g., a file) and every subject (e.g., a process) be linked to a security context. For files, the file context is stored in the form of extended attributes. The security information is thus directly associated with the file and is independent of the name of the file. The easiest way to determine the security context of a file is to use `ls -Z`. Alternatively, `getfattr -n security.selinux -d filename` also works:

```
user$ ls -Z /usr/sbin/httpd
system_u:object_r:httpd_exec_t:s0    /usr/sbin/httpd

user$ ls -Z /etc/httpd/conf/httpd.conf
system_u:object_r:httpd_config_t:s0 /etc/httpd/conf/httpd.conf
user$ getfattr -n security.selinux -d /etc/httpd/conf/httpd.conf
getfattr: Removing leading '/' from absolute path names
# file: etc/httpd/conf/httpd.conf
security.selinux="system_u:object_r:httpd_config_t:s0"
```

chcon and restorecon

SELinux is highly dependent on the fact that the correct context is stored for all files. To make this work when you create new files after installation, there are SELinux rules for many directories that automatically assign the appropriate context to the new files created in them. If this automatism fails, for example when moving files from another directory, you can correct or reset the context.

Provided your files are in the directories provided by SELinux, `restorecon` is the fastest way to get there. The following command correctly sets the context of all files stored in the `DocumentRoot` directory of Apache:

```
root# restorecon -R -v /var/www/html/*
```

If you have saved files in a different location, such as HTML files in the `/var/myotherserver` directory, you must use `chcon` to set the correct context:

```
root# chcon -R system_u:object_r:httpd_sys_content_t:s0 \
  /var/myotherserver
```

However, this method has the disadvantage that a later `restorecon` run for this directory would deactivate the set context again. A permanent solution is therefore to set up a new context mapping rule for the entire directory using the `semanage` command and then to adjust the context of all files in this directory to the new rule using `restorecon`:

```
root# semanage fcontext --add -t httpd_sys_content_t \
  "/var/myotherserver(/.*)?"
root# restorecon -R -v /var/myotherserver
```

But one question remains: How do you know what context is required? Answers are provided by the `man` pages from the `selinux-policy-doc` package. For each service monitored by SELinux, this package contains a page whose name is composed of `service_selinux`. All special rules that apply to the Apache web server can therefore be read via `man httpd_selinux`.

A lot of information on the correct use of `semanage` and `restorecon` is also available in an article on Stack Exchange, found at *https://unix.stackexchange.com/a/392834*.

Process context (domain)

For processes, the context is often referred to as the *domain*. You can determine the security context of a process (a domain) via `ps axZ`. As a rule, a process adopts the context of the account from which it is started. However, the context can also be changed automatically after startup by an SELinux rule. This is necessary if a specific program (like Firefox) is to be given a specific context regardless of who starts it or how it is started:

```
user$ ps axZ | grep httpd
unconfined_u:system_r:httpd_t:s0 2373 ?    Ss     0:00 /usr/sbin/httpd
unconfined_u:system_r:httpd_t:s0 2376 ?    S      0:00 /usr/sbin/httpd
...
```

The security context consists of three or four parts separated by colons: *user:role:type:mls-component*. Most importantly, the third part specifies the type of the file or process. Most SELinux rules analyze this information.

A detailed description of all four parts of the security context can be found at *https://fedoraproject.org/wiki/Security_context*.

Rules

The general syntax of a typical SELinux rule looks as follows:

```
allow type1_t type2_t:class { operations };
```

Here's a concrete example. The following rule allows processes whose context type is `httpd_t` to create new files in directories with context type `httpd_log_t`:

```
allow httpd_t httpd_log_t:dir create;
```

SELinux policies

A typical SELinux policy consists of tens of thousands of such rules! For speed reasons, SELinux does not expect the rules as text, but in a binary format. In an analogy to programming, one can also speak of a *compilation*. In current Fedora and RHEL versions, the `targeted` policy is used by default (`selinux-policy-targeted` package). It monitors selected programs and server services and is documented in the countless `man` pages of the `selinux-policy-doc` package.

Alternatively, you can install the *multilevel security* (MLS) policy designed specifically for servers. It is located in the *selinux-policy-mls* package. Based on this policy, RHEL achieves EAL 4 certification. This certification is required in the US for certain applications, often including military ones.

SELinux parameters (Booleans)

By now, you probably realize that making changes to the policy is difficult. To allow a certain degree of customization even without rule changes, the `targeted` policy contains various Boolean parameters that you can change at runtime. On Fedora or RHEL, the easiest way to do this is to use the `system-config-selinux` graphical user interface (see Figure 30.1), which is hidden in the `policycoreutils-gui` package. This package is not installed by default.

You can also use `getsebool` to read the value of Boolean configuration parameters. `setsebool` changes such parameters and, in the following example, allows Apache to run CGI scripts:

```
root# setsebool -P httpd_enable_cgi 1
```

You can get a list of all Boolean configuration parameters using `getsebool -a`. The last time I tested this on Fedora 38, they numbered almost 360:

```
root# getsebool -a
  abrt_anon_write - > off
  abrt_handle_event - > off
```

```
abrt_upload_watch_anon_write - > on
antivirus_can_scan_system - > off
...
```

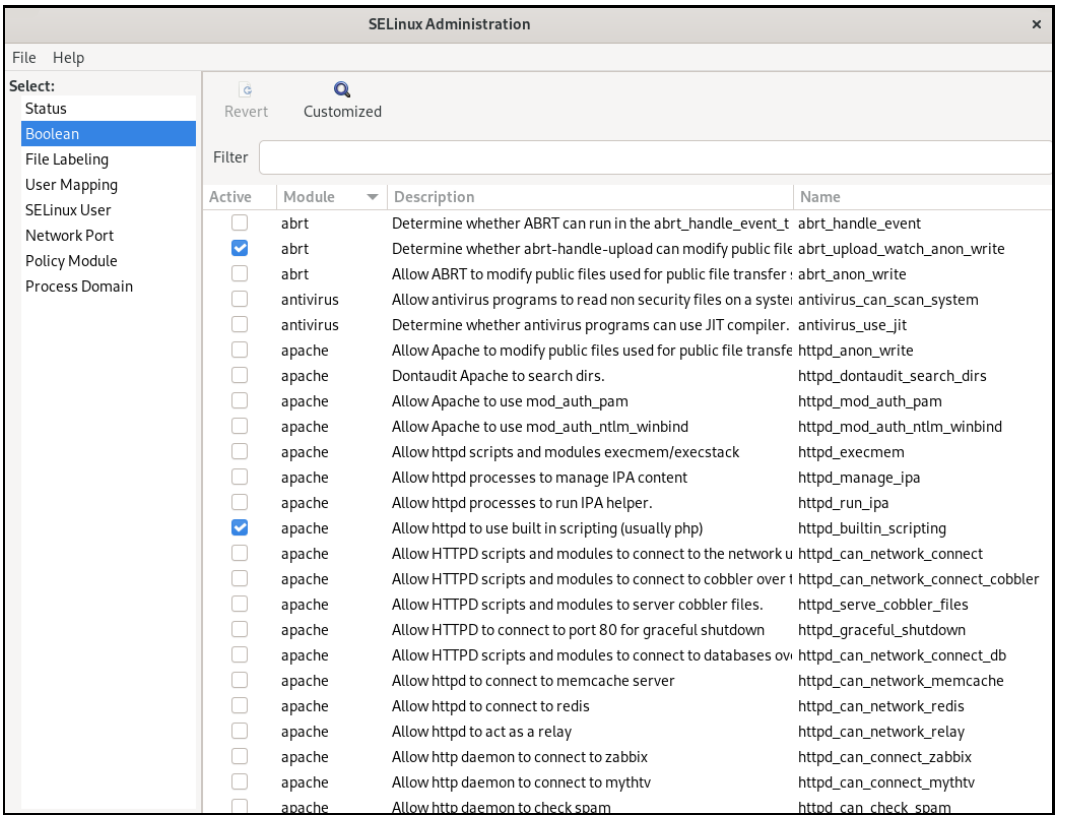


Figure 30.1 Changing SELinux Boolean Parameters

Documentation

The rules and Boolean parameters for various network services are each documented in their own `man` pages, but these must be installed separately:

```
root# dnf install selinux-policy-doc
```

You can then read the documentation via `man service_selinux`, replacing `service` with the respective service name. `man httpd_selinux` thus provides information about the SELinux rules that apply to the Apache web server, `man sshd_selinux` describes the rules for the SSH daemon, and so on. The following page provides a useful summary of the meaning of many Boolean parameters: *https://wiki.centos.org/TipsAndTricks(2f)SelinuxBooleans.html*.

Start and configuration

SELinux is implemented as part of the kernel. An explicit start by the init system is therefore not necessary. Likewise, there is no SELinux daemon or other background process.

The configuration is done by the files in the /etc/selinux directory. Of central importance is /etc/selinux/config. The file specifies which mode SELinux is running in (*enforcing*, *permissive*, or *disabled*) and which policy applies. However, changes to this file will only take effect after a restart:

```
# /etc/selinux/config
SELINUX=enforcing
SELINUXTYPE=targeted
```

Status

sestatus determines the current status of SELinux. SELinux with the *targeted* policy is active on the test computer:

```
root# sestatus
SELinux status:           enabled
SELinuxfs mount:          /sys/fs/selinux
SELinux root directory:   /etc/selinux
Loaded policy name:       targeted
Current mode:             enforcing
...
```

Fixing SELinux issues

Basically, the following options exist for responding to SELinux rule violations:

- You can search for a suitable Boolean parameter that matches your problem in the policy and set it correctly using system-config-selinux or setsebool.
- You can change the context information of the affected files.
- You can change or extend the policy. However, this requires much more SELinux knowledge than is provided in this section.
- You can turn SELinux off completely.

Problems caused by SELinux are not always visible at first glance. For example, if you copy a directory tree with HTML files to the /var/www/html directory using cp -a, then Apache will not be able to read the HTML files. Here's why: the cp option -a causes the EA and thus the SELinux context information to be copied as well. This circumstance prevents the copied files in /var/www/html from automatically receiving the correct context information through a SELinux rule. You can avoid these problems by using the cp -r variant instead of cp -a.

Apache itself does not know SELinux. The program only notices that it cannot access the files. In /var/log/httpd/error_log, you will then find an entry like the one in the following listing:

```
... [core:error] ... Permission denied: [
client 192.168.122.1:57032] AH00132:
file permissions deny server access: /var/www/html/test.txt
```

There is also no reference to SELinux in the journal. If your instinct still tells you that the issue might have to do with SELinux, you should install the `setroubleshoot-server` package. This package contains the `sealert` command, which helps to analyze the `/var/log/audit/audit.log` logging file. When starting `sealert`, you should make sure the right language settings have been set:

```
root# LANG= sealert -a /var/log/audit/audit.log
found 1 alerts in /var/log/audit/audit.log
SELinux is preventing httpd from read access on the file test.txt.

If you want to allow httpd to read user content
Then you must tell SELinux about this by enabling the
'httpd_read_user_content' boolean.
```

The proposed solution is not wrong. However, it's easier to set the context information of the affected files correctly by using `restorecon`:

```
root# restorecon -R -v /var/www/html/*
```

Disabling SELinux

In the past, you could disable SELinux using the `SELINUX=disabled` setting in the `/etc/selinux/config` file. But in current Fedora versions, as well as in RHEL from version 9, on this does not work anymore!

Furthermore, the `SELINUX=permissive` setting is permitted in `/etc/selinux/config`. This keeps SELinux running and logs rule violations to `/var/log/messages` or the journal. However, SELinux allows the rule violation and does not block the affected program. The `setenforce 0` command has the same effect.

If you want to disable SELinux altogether, you must pass the `selinux=0` option to the kernel. For a permanent shutdown, you need to include the option into the GRUB configuration. The easiest way to do that is as follows:

```
root# grubby --update-kernel ALL --args selinux=0
```

A complete deactivation is only recommended if you do not want to use SELinux in the future. Here's why: If SELinux is deactivated, all rules that assign the SELinux context information to new files are also deactivated. If SELinux is to be reactivated later, the files with missing context information will cause problems. For this reason, when the boot process detects a reactivation, relabeling is performed at the next boot to restore the context

information as best as possible. Afterward, another reboot will be carried out automatically.

Arch Linux, Debian, and SUSE

Most Linux distributions deliver a SELinux-compatible kernel. To activate SELinux, it is sufficient to install a few additional packages. Instructions for Arch Linux, Debian, and SUSE can be found on the following websites:

- *https://wiki.archlinux.org/title/SELinux*
- *https://wiki.debian.org/SELinux/Setup*
- *https://debian-handbook.info/browse/de-DE/stable/sect.selinux.html*
- *https://documentation.suse.com/sles/15-SP1/html/SLES-all/cha-selinux.html*
- *https://doc.opensuse.org/documentation/leap/security/html/book-security/cha-selinux.html*
- *https://doc.opensuse.org/documentation/leap/security/html/book-security/*
- *https://doc.opensuse.org/documentation/leap/security/html/book-security/cha-selinux.html*

Remember that you must deactivate the AppArmor system running by default on Debian and SUSE before activating SELinux.

30.2 AppArmor

Instead of adapting the complex SELinux system for its own distributions, Novell bought the Immunix company in 2005, renamed its *Subdomain* security solution to *AppArmor*, placed it under the GPL, and developed some YaST modules for administration for SUSE. A few years later, AppArmor was officially integrated into the kernel.

AppArmor is used by default by Debian (from version 10), SUSE, and Ubuntu. In recent years, developers employed by Canonical have been responsible for the continued development of AppArmor.

Like SELinux, AppArmor is a MAC security system. But in contrast to SELinux, AppArmor rules are based on absolute filenames. For this reason, separate tagging of all files via EAs is not necessary; moreover, AppArmor also works for file systems that do not support EAs. Wildcards are allowed in the AppArmor rules. For this reason, AppArmor gets by with far fewer rules than SELinux for typical use cases.

Criticism

However, there are also arguments against AppArmor:

- Security experts at Red Hat believe that absolute paths in rules are an inherent security risk. AppArmor's protection can be circumvented by

renaming files or directories, which of course only works if an attacker already has sufficient rights to do so.

- The policy for AppArmor is not as comprehensive as that of SELinux. Fewer programs are protected by default. While it is easier to create or change rules yourself than with SELinux, this kind of do-it-yourself security leaves a less than professional impression.

Links This section only provides an introduction to AppArmor. You can find information on this topic at the following pages:

- *https://documentation.suse.com/en-us/sles/15-SP1/* (see the Security Guide)
- *https://wiki.ubuntu.com/AppArmor*
- *https://wiki.ubuntu.com/SecurityTeam/KnowledgeBase/AppArmorProfiles*

30.2.1 AppArmor on Debian and Ubuntu

The following descriptions refer to AppArmor as it is active on Debian (from version 10) and Ubuntu. A short note about the SUSE-specific YaST module follows at the end of the chapter. By default, AppArmor is integrated into the kernel on Debian and Ubuntu. The security system is started by systemd, takes into account the basic configuration from the `/etc/apparmor` directory, and loads all rule files from the `/etc/apparmor.d` directory.

When AppArmor is started, the `securityfs` file system is mounted in the `/sys/kernel/security` directory. Its files provide information about active profiles, the number of rule violations that occurred, and so on.

Status The `aa-status` command provides an overview of the current state of AppArmor. The command returns both a list of all profiles and a list of actually monitored processes. The following list shows which services of an Ubuntu root server are monitored:

```
root# aa-status
apparmor module is loaded.
27 profiles are loaded.
27 profiles are in enforce mode.
   /snap/core/15511/usr/lib/snapd/snap-confine
   /snap/core/15511/usr/lib/snapd/snap-confine/ \
     /mount-namespace-capture-helper
   /usr/bin/freshclam
   /usr/bin/man
   /usr/lib/NetworkManager/nm-dhcp-client.action
```

```
   ...
4 processes are in enforce mode.
   /usr/bin/freshclam (1075)
   /usr/sbin/mysqld (873)
   /usr/sbin/mysqld (1789) docker-default
   /snap/canonical-livepatch/235/canonical-livepatchd (800) ...
```

In summary, this means that on the server, only four services are actually under the control of AppArmor. There are various other AppArmor profiles, but as the programs in question are not running, these profiles remain ineffective. Conversely, there are of course various other server services running on the server that should be monitored (Apache, Dovecot, Postfix, SpamAssassin, an SSH server)—but there are no official AppArmor rule profiles for those. Therefore, compared to an SELinux system, the protection is pathetic.

The picture changes a bit when you compare AppArmor and SELinux on a desktop system. There you can see that AppArmor secures various desktop programs, while SELinux focuses primarily on system and server services.

Debian versus Ubuntu

AppArmor is used on both Debian and Ubuntu. However, there are currently fewer rule profiles in Debian.

Rules (profiles)

The effect of AppArmor depends on the monitoring rules. These rules are also referred to as *profiles* and are located in the files of the `/etc/apparmor.d/` directory. For example, the `usr.sbin.cupsd` file contains the profiles for CUPS and the CUPS PDF driver.

The officially maintained rule profiles are usually provided by the respective package. Thus, the `user.sbin.cupsd` rule profile is available only if you have installed the CUPS printer system. For this reason, `/etc/apparmor.d` is initially almost empty after an Ubuntu server installation and only fills up to the extent that you install server services.

You can also install the `apparmor-profiles` package from the `universe` package source. On Debian, the package name is `apparmor-profiles-extra`. This package contains numerous other profiles that are not officially supported and maintained. Most profiles only run in the `complain` mode. In this mode, rule violations are logged but not penalized. You can use the `aa-enforce` and `aa-complain` commands from the `apparmor-utils` package to change the mode of a profile. To do so, you need to pass the full path of the program to be monitored to the two commands:

```
root# aa-enforce /usr/sbin/dnsmasq
Setting /usr/sbin/dnsmasq to enforce mode.
root# aa-complain /usr/sbin/dnsmasq
Setting /usr/sbin/dnsmasq to complain mode.
```

Alternatively, you can pass `aa-enforce` and `aa-complain` the filenames of the profile files. This makes it easy to change the mode of multiple profiles at once:

```
root# cd /etc/apparmor.d
root# aa-enforce usr.lib.dovecot*
```

For server services, you must also restart the respective program after activating an AppArmor profile:

```
root# systemctl restart <name>
```

Of course, profiles are only relevant independently of the AppArmor mode if the respective program is actually running. The `aa-status` command mentioned previously tells you which programs are currently being actively monitored.

Structure of rule files

Rules files, which are referred to as *profiles* in AppArmor, are in a simple text format. The following lines reflect the rules file for `mysqld`. Compared to SELinux, AppArmor rules are straightforward and easy to follow:

```
# File /etc/apparmor.d/usr.sbin.mysqld
#include <tunables/global>

/usr/sbin/mysqld {
  #include <abstractions/base>
  #include <abstractions/nameservice>
  #include <abstractions/user-tmp>
  #include <abstractions/mysql>
  #include <abstractions/winbind>

  # Allow system resource access
  /proc/*/status r,
  /sys/devices/system/cpu/ r,
  /sys/devices/system/node/ r,
  /sys/devices/system/node/** r,
  capability sys_resource,
  capability dac_override,
  capability dac_read_search,
  capability setuid,
  capability setgid,
```

```
# Allow network access
network tcp,
/etc/hosts.allow r,
/etc/hosts.deny r,

# Allow config access
/etc/mysql/** r,

# Allow pid, socket, socket lock file access
/var/run/mysqld/mysqld.pid rw,
/var/run/mysqld/mysqld.sock rw,
/var/run/mysqld/mysqld.sock.lock rw,
/var/run/mysqld/mysqlx.sock rw,
/var/run/mysqld/mysqlx.sock.lock rw,
/run/mysqld/mysqld.pid rw,
/run/mysqld/mysqld.sock rw,
/run/mysqld/mysqld.sock.lock rw,
/run/mysqld/mysqlx.sock rw,
/run/mysqld/mysqlx.sock.lock rw,

# Allow systemd notify messages
/{,var/}run/systemd/notify w,

# Allow execution of server binary
/usr/sbin/mysqld mr,
/usr/sbin/mysqld-debug mr,

# Allow plugin access
/usr/lib/mysql/plugin/ r,
/usr/lib/mysql/plugin/*.so* mr,

# Allow error msg and charset access
/usr/share/mysql/ r,
/usr/share/mysql/** r,

# Allow data dir access
/var/lib/mysql/ r,
/var/lib/mysql/** rwk,

# Allow data files dir access
/var/lib/mysql-files/ r,
/var/lib/mysql-files/** rwk,

# Allow keyring dir access
/var/lib/mysql-keyring/ r,
```

```
  /var/lib/mysql-keyring/** rwk,

  # Allow log file access
  /var/log/mysql.err rw,
  /var/log/mysql.log rw,
  /var/log/mysql/ r,
  /var/log/mysql/** rw,

  # Allow read access to mecab files
  /var/lib/mecab/dic/ipadic-utf8/** r,

  # Allow read access to OpenSSL config
  /etc/ssl/openssl.cnf r,
  # Site-specific additions and overrides. See local/README for details.
  #include <local/usr.sbin.mysqld>
}
```

Abbreviation	Meaning
r	Allows read access (read).
w	Allows write access (write).
a	Allows to expand the file (append).
l	Applies the same rules to hard links as to the source file (link).
k	Allows to block (lock) the file.
m	Allows the mmap function (allow executable mapping).
ix	The program inherits the rules of the base program (inherit execute).
px	The program has its own AppArmor profile (discrete profile execute).
ux	Executes the program without AppArmor rules (unconstrained execute).

Table 30.1 Basic AppArmor Access Rights

In the rules files, some include files are read first before basic features of the program are defined (see `man capabilities`). The other rules specify which files the program may use and how.

In AppArmor rules files, the wildcard character * is a placeholder for any number of characters. ** has a similar meaning, but includes the / character and thus includes files in all subdirectories.

Access rights are expressed by letters or letter combinations, the meanings of which are described in detail in the AppArmor Administration Manual. Table 30.1 explains the most important letters. The `?x` combinations control the rights of subprocesses that are started by the main program.

Rule parameters (tunables)

AppArmor provides a mechanism to change individual parameters of the rules in a simple way. These parameters are defined in the files of the `/etc/apparmor.d/tunables` directory. In the current implementation, however, there are only a few parameters that allow you to customize the location of the home directories, for example. If your server provides other locations for home directories besides `/home`, you must modify the `@{HOMEDIRS}` variable. In the following example, two additional directories, `/home1` and `/myhome`, were added there:

```
# File /etc/apparmor.d/tunables/home
@{HOME}=@{HOMEDIRS}/*/ /root/
@{HOMEDIRS}=/home/ /home1/ /myhome/
```

Logging and maintenance

Details about rule violations that occurred in the `complain` or `enforce` mode are passed as kernel messages and are recorded by default in the `/var/log/kern.log` and `/var/log/syslog` files. You can recognize AppArmor messages by the `audit` keyword:

```
root# grep audit /var/log/kern.log
[...] audit(1238580174.435:3): type=1503 \
          operation="inode_permission"
        requested_mask="a::" denied_mask="a::" name="/dev/tty"
        pid=6345 profile="/usr/sbin/cupsd" namespace="default"
[...] audit(1238580174.435:4): type=1503 \
          operation="inode_permission"
        requested_mask="w::" denied_mask="w::" name="/etc/krb5.conf"
        pid=6345 profile="/usr/sbin/cupsd" namespace="default"
```

The audit messages are often an indicator that the AppArmor rules are incomplete. A misbehavior of the program is of course also possible, but rather unlikely. Only an expert for the respective program can judge this with certainty. In this respect, an appropriate response to rule violations is difficult.

If you suspect that the affected program is working properly, you should switch the profile to the `complain` mode and report the audit message in the Ubuntu bug system (*https://bugs.launchpad.net*). You can also try to add a rule to the profile that allows the reported operation. Or you can simply ignore the message, so long as the affected program continues to run without any problems.

30.2.2 AppArmor on SUSE

Configuration and start

AppArmor is installed by default in current openSUSE and SUSE Enterprise distributions and can be started using systemd. YaST contains a module for the AppArmor configuration (see Figure 30.2). However, its functionality does not go far beyond that of the `aa-status`, `aa-enforce`, and `aa-complain` commands.

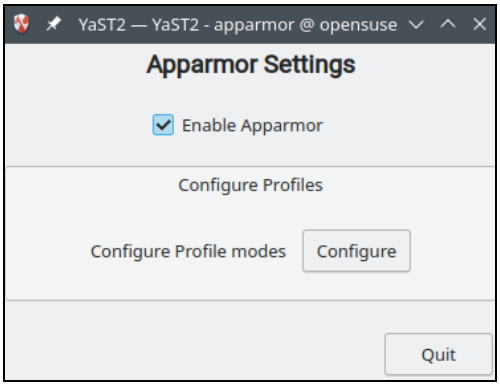


Figure 30.2 Unambitious YaST Module for AppArmor Configuration

PART VIII

Virtualization

Chapter 31
VirtualBox

Virtualization makes it possible to run several operating systems in parallel on one computer. This results in countless applications: you can try Linux on Windows or run Windows on Linux, safely test a new trial version of a distribution without compromising the existing Linux installation, safely separate server functions, and more.

The VirtualBox program available for the Windows, Linux, and macOS platforms (only for Intel CPUs) is best suited for desktop virtualization—that is, for running virtual machines that are supposed to be operated in the graphics system. German company InnoTek was originally behind VirtualBox. In 2008, Sun acquired InnoTek, and in 2010, Oracle bought Sun. This means that Oracle is now the owner of VirtualBox. Comprehensive documentation on VirtualBox can be found at *https://virtualbox.org*.

Large parts of VirtualBox consist of open-source code. The only exceptions are some additional functions that need to be installed separately. Their use is free of charge for private users only.

This chapter describes how you can install VirtualBox on Linux and run virtual machines on it. To a limited extent, VirtualBox is also suitable for server virtualization. However, the KVM program, which I will introduce in more detail in Chapter 32, is better suited for this purpose.

VirtualBox versus KVM

My recommendation to prefer VirtualBox for desktop use and KVM for server use is not without restrictions; personally, I also use KVM for desktop virtualization. This has to do with the fact that the somewhat awkward operation of the Virtual Machine Manager (see Chapter 32, Section 32.2) does not bother me. Rather, I enjoy the advantages of KVM over VirtualBox. For one thing, it eliminates the hassle of VirtualBox drivers on the host and in guests. For another, virtual machines continue to run when I close a window or when I log out and log in again.

31.1 Installing VirtualBox

There are basically two ways to install VirtualBox. The convenient variant is to simply use the VirtualBox packages that are in the package sources of

your distribution. If these packages are missing or not sufficiently up-to-date, you should download VirtualBox from *https://virtualbox.org* and install the program manually, which is a little more complex.

Aside from this, there is still some preparatory work to be done once the installation is complete, which I will explain at the end of this section.

Host and Guest

When describing virtualization systems, it has become common to refer to the basic system as the *host* and the virtual machines running on it as *guests*. In this chapter, I assume that the host is an already functioning Linux system.

Why the Latest Version?

Does it matter whether you are working with VirtualBox 6.1.28 or 7.0.10? In fact, neither the operation nor the basic functions of VirtualBox have changed much in recent years. So from a purely visual point of view, you won't notice much of a difference. Installing current versions is still worthwhile because only they ensure compatibility with current kernel versions and graphics drivers. So if you want to try new Linux distributions, VirtualBox must also be up to date.

31.1.1 VirtualBox Packages of your Distribution

Installing VirtualBox on Linux

Most distributions provide ready-made VirtualBox packages. The package name is usually `virtualbox`.

On Fedora, you must enable the `rpmfusion` package source beforehand; the package name is `VirtualBox`. (Pay attention to the capitalization!). In openSUSE, the core functions and the user interface are in separate packages. There you install the `virtualbox-qt` package. However, the various `virtualbox-guest` packages are not required! These packages contain drivers to be run *in* virtual machines—that is, when a Linux distribution itself is to be run inside VirtualBox.

VirtualBox kernel modules

VirtualBox accesses the `vboxdrv`, `vboxnetadp`, and `vboxnetflt` kernel modules on the host system. Some distributions provide these modules in binary format through a separate package that is updated with each kernel update. On openSUSE, the package name is `virtualbox-kmp-default`.

DKMS

In other distributions, the source code of the VirtualBox packages is installed. With each kernel update, the corresponding VirtualBox modules must be recompiled. This is taken care of by Dynamic Kernel Module Support (DKMS) in some distributions. This is the case with Ubuntu, for example. There, the `virtualbox-dkms` package is automatically installed with the rest of the VirtualBox packages.

akmods (Fedora)

The RPMFusion package source for Fedora provides the `akmods` command instead of DKMS. If you have just performed a kernel update, you must restart the computer first. Then enable the RPMFusion package sources and run the following commands:

```
root# dnf install VirtualBox akmod-VirtualBox
root# akmods
root# systemctl restart systemd-modules-load
```

The first command installs the required packages, the second one compiles the kernel modules, and the third command loads them. From then on, `akmods` will take care of updating the VirtualBox drivers on its own after each kernel update.

Test

To check whether the compilation and loading of the VirtualBox kernel modules worked, you should use the following command:

```
root# lsmod | grep vbox
vboxnetadp             28672  0
vboxnetflt             28672  0
vboxdrv               516096  2 vboxnetadp,vboxnetflt
```

31.1.2 VirtualBox Packages from Oracle

Instead of the VirtualBox packages provided with your distribution, you can also install the version offered for download by Oracle at *https://virtualbox.org/wiki/Linux_Downloads*. This is especially useful if Oracle offers a newer VirtualBox version than your distribution. On that website, you can find VirtualBox in various formats: as RPM and Debian packages for various distributions, as a universal installer, and as package sources.

At this point, I used to recommend setting up the respective package source because you can get updates from it in an uncomplicated way. Recently, however, Oracle has only poorly maintained these package sources. At the time of my tests in July 2023, there was no package source at all available for the then current Ubuntu version 23.04. The package source for Ubuntu

22.04 again contained only the outdated VirtualBox version 6.1. Thus, it currently makes much more sense to download the appropriate package for your distribution and install it manually without modifying any package sources. You may need to repeat this process at a later time if you need a VirtualBox update.

Universal installer

You should only use the installation program if the other installation methods are not possible. For this purpose, you need to run the following commands after downloading:

```
root# chmod u+x VirtualBox_nnn.run
root# ./VirtualBox_nnn.run install
```

Kernel modules

On Debian and Ubuntu, DKMS takes care of automatically recompiling all modules required for VirtualBox during kernel updates. If DKMS is not available or fails, you must recompile the kernel modules yourself using one of the following two commands:

```
root# /usr/lib/virtualbox/vboxdrv.sh setup
root# /sbin/vboxconfig
```

However, the `gcc` C compiler and the kernel header files are also required for compilation. Many distributions require you to install the appropriate packages up front (see Chapter 21, Section 21.3).

31.1.3 Preparation Tasks

Storage location for virtual machines

VirtualBox sets up a subdirectory within `VirtualBox VMs` for each virtual machine. The virtual machine settings and the virtual hard disk are stored there in several files. If necessary, you can set a different location via **File • Settings**.

Extension pack

Oracle offers an extension pack for download on its website. When downloading the extension pack, the web browser suggests opening the file directly using VirtualBox. You can accept this suggestion. The extension pack adds some additional features to VirtualBox: among other things, you can then access USB devices (USB 2 and USB 3), PCI cards, and webcams in the virtual machines and control the virtual machines via Remote Display Protocol (RDP) on another computer in the network. These extensions are distributed in binary format only, so they are not open-source code. The commercial use of these extensions requires a license from Oracle!

vboxusers group

Regardless of the source of your VirtualBox installation, the `vboxusers` group has been created. Only users that belong to this group can access USB devices in virtual machines. You should therefore add your account to the

vboxusers group before starting VirtualBox for the first time. Replace kofler with your login name in the following command:

```
root# usermod -a -G vboxusers kofler
```

For the changed group assignment to take effect, you must log out and log in again.

31.1.4 Installing VirtualBox on Windows or macOS

Installing VirtualBox on Windows or macOS (Intel CPUs only) is basically a breeze: you must download the appropriate setup program from the VirtualBox page and run it. If you fail to start VirtualBox machines on your Windows computer (the most common error code is VERR_NEM_VM_CREATE_FAILED), you should first take a look at the EFI/BIOS settings of your computer. Furthermore, the virtualization functions of the CPU must be enabled.

Hyper-V

Interaction with Microsoft's own Hyper-V virtualization technology also used to be a problem in the past. There are many recommendations on the internet to simply deactivate Hyper-V. However, as more and more Windows functions require Hyper-V, this is rarely a good idea. Since version 6, VirtualBox can fortunately make use of Windows's own virtualization functions. After massive initial stability problems, this feature has worked well in my tests recently.

Apple Silicon

Since version 7, there is a beta version of VirtualBox for Macs with Apple's own CPUs (M1, M2, etc.). However, when I last tried this version, I encountered many problems. It will probably take some time before productive use is possible.

A free alternative is the UTM program (*https://mac.getutm.app*), which is internally based on QEMU like KVM. You can use it to run Linux distributions compiled for ARM CPUs. However, the UTM user interface takes some getting used to and offers far fewer configuration options than other virtualization programs.

31.2 Setting Up VirtualBox Machines

Once VirtualBox has been installed, you can start setting up virtual machines. A graphical user interface (see Figure 31.1) helps with both configuration and subsequent changes to settings.

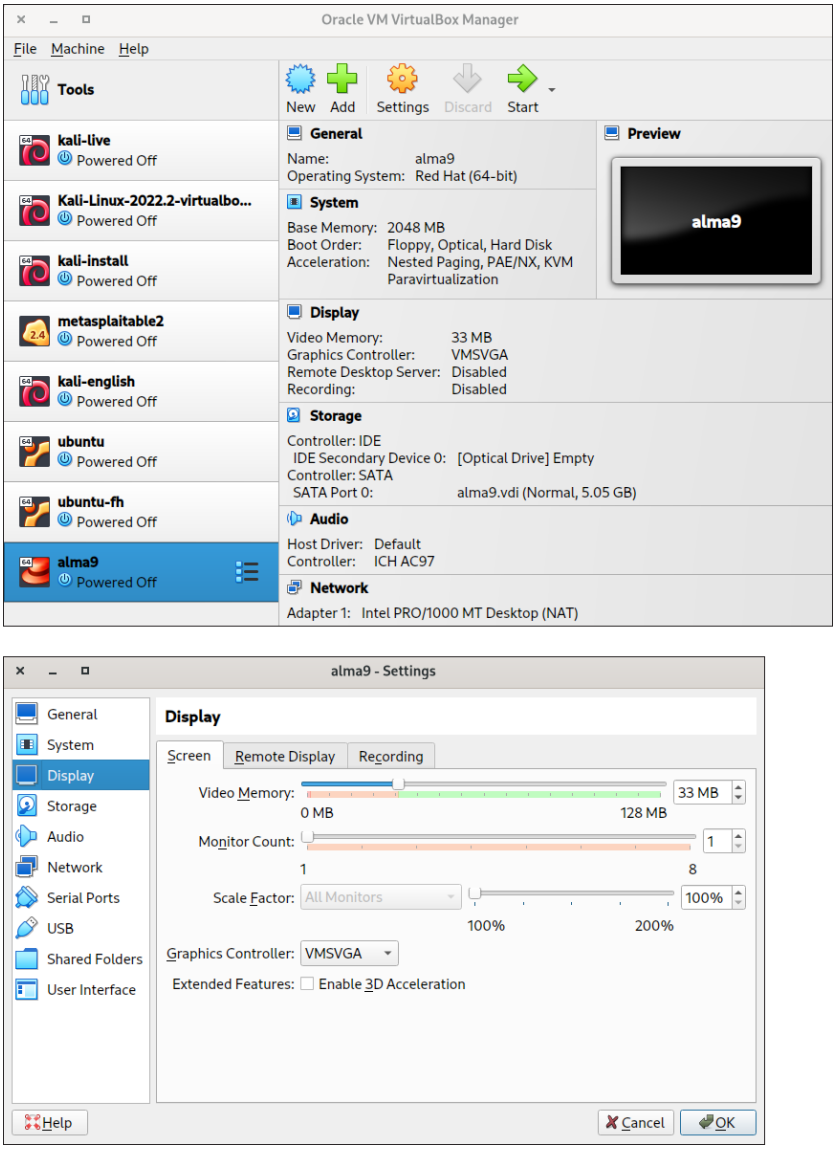


Figure 31.1 Overview of All Virtual Machines (Top) and Their Settings (Bottom)

31.2.1 Setting Up a Linux Virtual Machine

When setting up a virtual machine, you will be assisted by a wizard that you can start by clicking the **New** button. (**Add**, on the other hand, is used to import an already existing virtual machine.)

The first step is to assign a name to the new virtual machine and select the ISO file with the installation media. In many cases, VirtualBox will then automatically detect the operating system; otherwise, you need to specify

Linux as the type and the distribution as the **Version**. If your distribution is not included in the long dropdown list, select **Linux 2.6 / 3.x / 4.x / 5.x**. This setting applies to all current kernel versions, including 6.x. Make sure that there are *two* versions for each operating system: one for 32-bit and one for 64-bit installations. Almost all common distributions require a 64-bit architecture.

Unattended Installation

In some distributions, VirtualBox provides a **Unattended Installation** for selection. If you use this, you specify the password and a few other parameters, and VirtualBox takes care of the rest. This saves you from having to use the distribution's installer and install the VirtualBox guest additions later.

That sounds tempting. Unfortunately, however, I have not had a good experience with this feature and advise you not to use it. The main problem is that VirtualBox does not care about the language settings. Depending on the distribution, VirtualBox also "forgets" the `sudo` configuration. You must use `su -l` to switch to `root` mode (using the same password as the default user) and then add your account to the `sudo` or `wheel` group.

Long story short, the time you gain with the unattended installation is lost again when you configure the distribution later.

Configuration

In the dialogs that open next, you need to specify the memory, the number of assigned CPU cores, and the size of the virtual hard disk. Modern distributions should be allocated at least 3 to 4 GiB of RAM, two CPU cores, and a disk size of around 20 GiB.

Finally, VirtualBox displays a summary of all hardware components. If necessary, you can later use the **Change** button to make further settings—for example, to change the network access or select a different ISO file as the data source for the DVD drive in the **Mass Storage** dialog sheet.

First launch

Once the configuration is complete, you can start the virtual machine. Only now does VirtualBox test whether the required kernel modules can be loaded and whether hardware virtualization functions are available. If one of these conditions is not met, an error message or warning will display.

Depending on the installation variant, it usually helps to recompile the VirtualBox kernel modules using the `/usr/lib[64]/virtualbox/vboxdrv.sh setup` script or the `/sbin/vboxconfig` command. You should also bear in mind that the hardware virtualization functions must be activated in the BIOS or EFI.

Another possible cause of the error can be that another virtualization system is running and blocking the CPU's virtualization functions (e.g., KVM/libvirt or the Android emulator of Android Studio). On Linux, only *one* virtualization system can be active at a time. Note that the Docker Desktop program also runs a virtual machine internally and can therefore cause a conflict.

Host keyboard shortcut

The virtual machine automatically gets the keyboard and mouse focus as soon as you press a key. By default, you can release the focus using the right Ctrl key. In the VirtualBox main window, you can set a different "Host" key via **File • Preferences • Input • Virtual Machine**. The currently valid combination is displayed on the right in the status bar of the VirtualBox window. The most important host key combinations are summarized in Table 31.1.

Shortcut	Meaning
Host	Release keyboard and mouse focus
Host + F	(De)activate full screen mode
Host + Del	Ctrl + Alt + Del send to guest system
Host + Backspace	Ctrl + Alt + Backspace send to guest system
Host + F#	Ctrl + Alt + F# send to guest system
Host + S	Create backup of a virtual machine
Host + H	Switch off virtual machine via ACPI
Host + R	Switch off virtual machine immediately (reset—be careful!)

Table 31.1 VirtualBox Keyboard Shortcuts

Graphics resolution

It often happens that virtual machines are initially only displayed with a graphics resolution of 800 × 600 or 1024 × 768 pixels. Ideally, the graphics resolution should adjust automatically when you resize the window—but depending on the distribution, this only works after installing the VirtualBox guest additions (and unfortunately not always even then).

The chances of success are higher if you simply open the settings program in the guest's desktop system. As a rule, you can select different preset resolutions there. Sometimes there is also a "crooked" resolution that corresponds exactly to the current window size.

Especially in older distributions, an alternative approach consists of passing the kernel option `vga=791` at startup. This activates a graphics mode

with a resolution of 1024 × 768 pixels, which is at least sufficient for the installation. However, the option is not recognized by every distribution.

Graphics Issues

If the graphics in the virtual machine cause problems, you should first increase the graphics memory in the virtual machine settings. By default, 33 MiB is usually reserved. A maximum of 128 MiB is possible and is also recommended for operating the virtual machine at a high resolution.

31.2.2 Installing Guest Additions

Once the actual installation is complete and you have restarted the virtual machine, you should first install any updates available for your distribution. After another reboot, it's recommended to install the so-called guest additions as well. These improve the interaction between host and guest systems. Mouse movements should work more smoothly, the virtual screen resolution of the guest automatically adapts to the window size, data can be exchanged with the host system via shared folders, text can be copied via the clipboard, and so on.

Some distributions ship ready-made packages with the VirtualBox guest additions. However, these packages are rarely up to date, so you may pay for the convenience of the installation in the form of incompatibilities with the more recent VirtualBox version you are using:

```
root# apt install virtualbox-guest-dkms \          (Debian, Ubuntu)
    virtualbox-guest-utils \
       virtualbox-guest-x11
root# dnf install VirtualBox-guest-additions       (Fedora with RPMfusion)
root# zypper install virtualbox-guest-tools \      (openSUSE)
    virtualbox-guest-tools
```

For other distributions, or if you need the latest version of the guest additions, you must perform a manual installation. To do this, eject any mounted CD/DVD and then run **Devices • Insert Guest Additions CD Image** in the VirtualBox window. Usually, after a few seconds, a file manager window appears in the virtual machine where you can launch `autorun.sh`. If this does not work, the following commands will help:

```
root# mkdir /mnt/cdrom
root# mount /dev/sr0 /mnt/cdrom
root# sh /mnt/cdrom/autorun.sh
```

The installer then sets up the `vboxguest` kernel module and the `VBoxClient` program and adds some scripts so that the guest additions are also used the next time the virtual machine gets started.

On Ubuntu, the installation of the guest additions works right away. Most other Linux distributions require you to install various packages that contain the C compiler and kernel header files before installing the guest additions. You should run an update and reboot the system beforehand to make sure that the installed kernel version and the version of the kernel header files match:

```
root# apt    install gcc make linux-headers-amd64          (Debian)
root# dnf    install kernel-devel.x86_64                   (Fedora)
root# zypper install gcc make kernel-source kernel-syms   (openSUSE)
```

If the first attempt to install the driver failed due to missing packages, you can repeat the installation using the following command:

```
root# /sbin/rcvboxadd quicksetup all
```

If problems still occur, the current kernel version may not match the version of the kernel code files. Then you should perform a full update, restart the virtual machine, and try again!

31.2.3 Setting Up a Windows Virtual Machine

Provided you have an installation CD/DVD or the corresponding ISO file as well as a valid license and the corresponding key, you can also install Windows in VirtualBox. Installing Windows and the VirtualBox guest additions went smoothly in my tests. The installation of Windows 11 requires VirtualBox to be at least version 7.0.

I recommend that you wait until you are happy with the service before registering online. If you later increase your RAM or change other virtual hardware parameters in the virtual machine settings, you may need to repeat the registration!

31.3 Working Techniques and Configuration Tips

This section provides tips on optimizing virtual machines and on data exchange between the host system or the "real" local network and the virtual machines.

31.3.1 Network Configuration

VirtualBox provides its guests with the host's network infrastructure in the form of a virtual network adapter. There are different methods of routing network traffic from the virtual network adapter to the real network. The corresponding parameters can be found in the settings dialog in the **Network** dialog sheet (see Figure 31.2). The decisive factor is the setting of the **Attached To** list box, where the default value is **NAT**:

- **NAT**
  In the NAT variant, VirtualBox provides its guests with its own DHCP server and implements masquerading (NAT). In this way, guests can use the internet access of the host system. Access to the local network is impossible because of the different address ranges for the local network and the virtual NAT network of the virtualization system. Likewise, you cannot establish an SSH connection from the host to the guest. The virtual machines are separated from the host as if by a simple firewall.

  In the NAT variant, VirtualBox uses the IP address 10.0.2.2 on the host. The virtual machines get different 10.0.2.* addresses.
- **Bridged adapter**
  In this variant, the guest appears as an additional client on the local network. This variant is ideal if there is a DHCP server in the local network or if the host computer is connected to an ADSL or Wi-Fi router. The virtual guests then obtain their network configuration via this server/router and can access both the local network and the internet. If your host computer has multiple network interfaces, you must specify which interface connects to the local network.

  In the office, this variant is my preferred configuration: real and virtual machines are thus equal partners on the local network, and data exchange via SSH, Samba, and the like works without a problem. Note, however, that the bridged adapter does not work in some (corporate) wireless networks. I have had the best experience with this configuration variant when the host computer is connected to the local network via an Ethernet cable (i.e. not via Wi-Fi).
- **Host-only adapter**
  In this variant, the guest can only communicate with the host via the network functions, but not with other computers on the local network or with the internet. This variant is useful if you want to build a test system that consists of several virtual machines and that is not accessible from the outside.

- **Internal network**
  Here VirtualBox creates a virtual network in which only virtual machines can communicate. With this variant, you have access neither to the local network nor to the internet.

You can equip virtual machines with up to four network adapters. This enables you to use several configuration variants in parallel—for example, a NAT adapter so that the virtual machines have internet access and a host-only adapter so that you can establish an SSH connection between the virtual machines and the host computer.

The network configuration can be changed during operation! Thus, it is not necessary to restart the virtual machine every time a change is made. The quickest way into the configuration dialog is via the **Network Adapter Activity** icon in the status bar of the VirtualBox window.

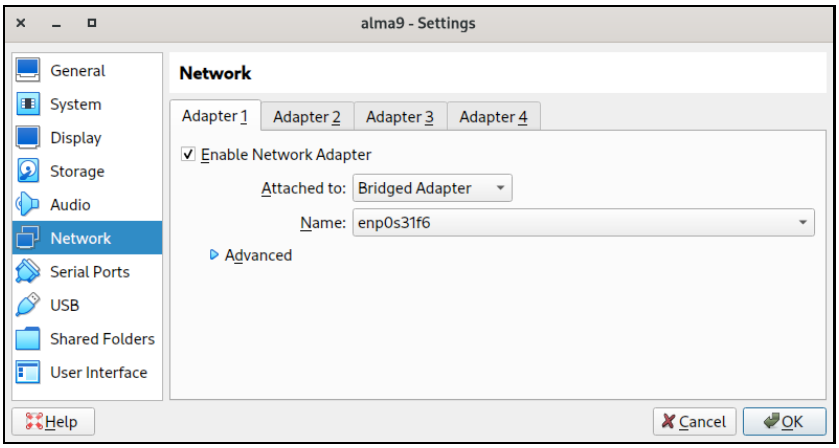


Figure 31.2 Network Settings for Virtual Machine

31.3.2 SSH Access via Port Forwarding

If the network adapter uses the default **NAT** setting, there is no way to establish an SSH connection from the host to the virtual machine. The simplest solution is port forwarding. To do this, you need to run **Devices • Network • Settings** in the VirtualBox window, expand the **Advanced** section of the adapter, and click **Port Forwarding** (see Figure 31.3). Then set up a new rule that connects port 2222 of the host (127.0.0.1) to port 22 of the virtual machine (10.0.2.15).

After you have saved the settings (it isn't necessary to restart the virtual machine), you can establish an SSH connection to the virtual machine in the terminal of the host computer using the following command:

```
user@hostsystem ssh -p 2222 username_in_vm@localhost
```

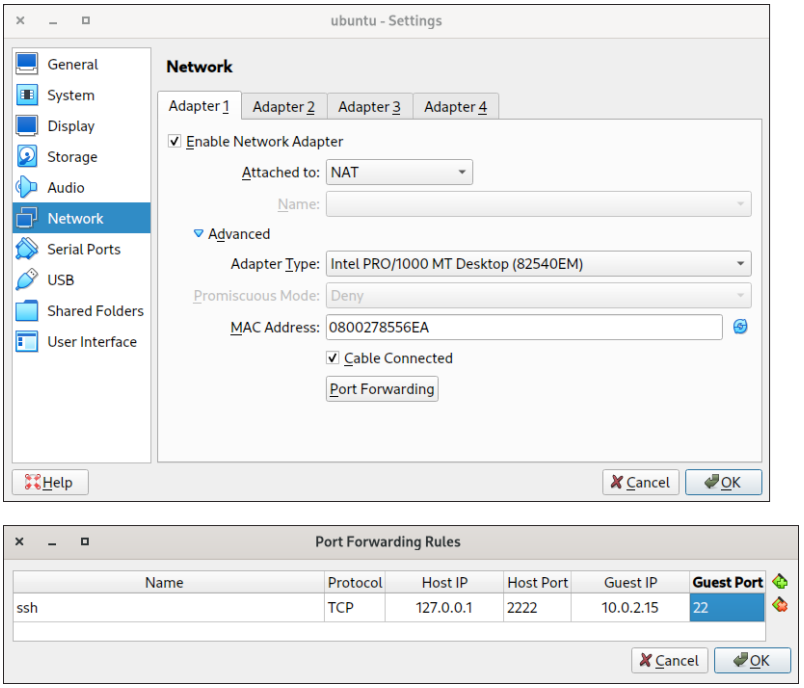


Figure 31.3 Port Forwarding for SSH

The `-p 2222` option is important here. `ssh` should not use port 22 as usual, but port 2222. It's also important that you specify `localhost` as the destination address. Because of the port forwarding, you end up in the virtual machine.

31.3.3 Data Exchange via the Clipboard

In the **General • Advanced** dialog sheet of the virtual machine settings, you can enable the **Bidirectional** mode for the shared clipboard and for the drag-and-drop feature. In any case, both options require the guest extensions to be installed in the virtual machine.

This configuration allows you to copy text between the host and the guest via the clipboard. In addition, you can now use drag and drop to copy files back and forth between a file manager in the host and a file manager in the guest. In my tests, this only worked occasionally; this function does not really seem mature.

31.3.4 Exchanging Data with a Shared Folder

Host configuration

Shared folders are a more reliable way of exchanging data between the host and guest. To configure this, open the settings dialog using **Settings**, switch

to the **Shared Folders** dialog, select a local directory on the host system, assign a name to the folder (e.g., `myshare`) and assign it a mount point (a mount directory, such as `/mnt/myshare`) within the virtual machine. The **Auto-mount** option causes the VirtualBox guest additions to take care of setting up the directory in the virtual machine.

Guest access

In a Linux virtual machine, you can find the data in the previously specified mount directory. In Windows guests, you can find the shared directory in Explorer as the network directory of the virtual machine—`vboxsrv`. If you have used the **Auto-mount** option during the configuration, then the directory will also be assigned its own drive letter on Windows.

31.3.5 USB Devices in Virtual Machines

Provided that you have installed the VirtualBox Extension Pack on the host, you can also use USB devices in virtual machines. This only works if the USB device is *not* used in the host system. USB media are usually automatically mounted in the file system in the host system; you must unmount them to use them in the guest.

Another requirement is that the user running VirtualBox is a member of the `vboxusers` group. Finally, you must make sure that the **USB Controller** is enabled in the virtual machine settings in the **USB** dialog sheet. In this dialog sheet, you can also define a filter to assign a USB device directly to a virtual machine. However, this is not a mandatory requirement. You can also dynamically associate the USB device with the virtual machine's USB icon in the VirtualBox status bar after power on.

31.3.6 Exporting/Importing Virtual Machines

To share a virtual machine, you can use **File • Export Appliance** in order to create a *virtual appliance*—that is, a virtual machine intended for sharing—which usually consists of two files: `*.ovf` contains a description of the virtual machine, and `*.vmdk` contains the hard disk image in compressed form. You can then set up this virtual machine again in another VirtualBox installation via **File • Import Appliance**.

Transferring a Virtual Machine to Another Host

If your only concern is to move one or more virtual machines from one machine to another, you can save yourself the trouble of converting to a virtual appliance. In this case, it's sufficient to copy the relevant `VirtualBox VMs/vm-name` directory. Then run the **Machine • Add** command in VirtualBox and select the `*.vbox` file.

31.3.7 Grouping Virtual Machines and Running Them Invisibly

Groups

If you set up multiple virtual machines in VirtualBox, you can group them via a context menu and move them by using drag and drop. Not only do groups provide a better overview in the list of virtual machines, but they also give you the option of starting or shutting down all virtual machines in a group together. This is particularly useful if the group is related in terms of content (e.g., to test a server with two clients).

Virtual machines without windows

Usually, each running virtual machine is displayed in its own window. When closing the window, you have the option of saving the status of the virtual machine (i.e., pausing the virtual machine), shutting it down via ACPI, or forcibly stopping it (such as by disconnecting a power cable).

Sometimes, however, it can be convenient to run virtual machines invisibly, *without* their own windows. This is especially true for server installations that run in text mode anyway. For such virtual machines, you can select the **Start** button and then the **Headless Start** menu item.

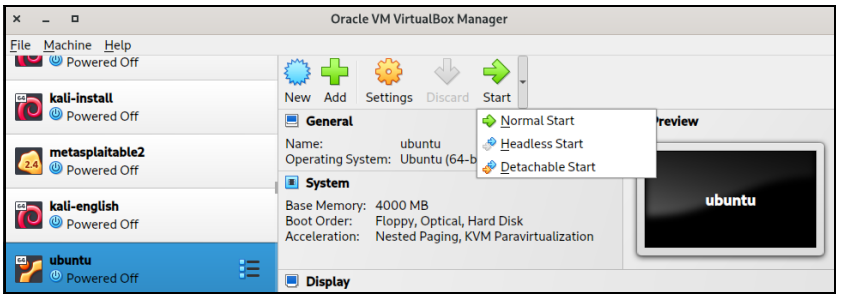


Figure 31.4 Start Button Menu

Even more flexibility is provided by the **Detachable Start** item (see Figure 31.4). This will display the virtual machine in a window at startup as usual. However, by using **Machine • Detach GUI**, you can then close the window if necessary without stopping the virtual machine. You need to double-click the virtual machine icon in the VirtualBox main window to revive the virtual machine interface if necessary.

Unfortunately, there is no way to preset in the virtual machine settings that this virtual machine should always start without a window.

31.3.8 Using VirtualBox with a High-Resolution Monitor

If you use VirtualBox on a host with a high-resolution monitor, the virtual machine display may be much too small. You have two options to change that:

- In the virtual machine properties, you can set a fixed scaling factor in the **Display** dialog.
- Alternatively, you can activate the scaled mode during operation by pressing the Host key and C. In this mode, the virtual screen gets adjusted to the window size and scaled accordingly. For reasons that are not entirely clear, the VirtualBox window loses its menus and status bar in this mode. Again Host+C turns the scaled mode off again, while the menus and status bar reappear.

However, the speed of the image buildup and the display quality suffer with both variants.

31.3.9 Controlling VirtualBox by Command (vboxmanage)

Instead of operating VirtualBox through its user interface, you can also perform a large number of operations through the `vboxmanage` command. I don't have enough space here to go into the countless subcommands and options. Instead, I will limit myself to a few possible applications in this and the following example. For a comprehensive reference, refer to the VirtualBox manual at *https://www.virtualbox.org/manual/UserManual.html#vboxmanage*.

Shutting down all virtual machines

Virtual machines, even if they appear to be idle, sometimes cause a considerable CPU load. Among other things, this is due to the many background services in the virtual machines that check for updates. If I need the CPU more urgently for other tasks or if I am simply bothered by the fan of my notebook, I can simply pause all virtual machines with the following script:

```
#!/bin/bash
vboxmanage list runningvms |  sed -r 's/.*{(.*)}/\1/' |
  xargs -L1 -I {} vboxmanage controlvm {} pause
```

`vboxmanage list runningvms` returns a list of all running virtual machines. `sed` extracts the ID numbers from it. `xargs` passes these to a second `vboxmanage` command and executes `pause`. To resume the execution of the virtual machines, there is a second, similar script—namely, `resume` instead of `pause`. Then I call a third script before I shut down the computer. This script sends the following ACPI shutdown signal to all virtual machines: `vboxmanage controlvm acpipowerbutton`.

31.3.10 Enlarging Virtual Hard Disks

Linux guest

The VirtualBox user interface does not provide the option of subsequently enlarging a virtual hard disk. The `vboxmanage` command again provides a

solution here. Before you get started, you need to shut down your virtual machine. In addition, a full backup is highly recommended!

Next, you need to locate the `*.vdi` file of the virtual hard disk and apply the `vboxmanage` command to it. Then you want to use the `--resize` option to specify the preferred new size in MiB. As a rule, the command is executed at lightning speed:

```
root# vboxmanage modifyhd debian.vdi --resize 60000
```

But that's only half the battle. The virtual machine does not yet know that its hard disk has become bigger. For a guest installation without LVM and with the `ext4` or `xfs` file system, you must now mount an ISO image of a Linux live CD in the virtual CD/DVD drive and start a live system inside the virtual machine. There you run `parted /dev/sda` and can then increase the size of the last partition. After that, you also need to increase the size of the file system it contains by using `resize2fs` or `xfs_growfs`.

On the other hand, if you use LVM or the `btrfs` file system in the Linux guest, you can increase the file system size on the fly. These interventions are, of course, not without risk. That's why you should read the relevant sections from Chapter 18 beforehand!

Windows guest

Similarly, you can also enlarge a Windows file system. In this case too you must first shut down the virtual machine and enlarge the `*.vdi` file using the `vboxmanage` command. Then start Windows, open a command prompt window in it with admin privileges, and run the following commands:

```
> Diskpart
list disk
select disk 0
list partition
select partition 2
extend
```

`list disk` returns a list of all virtual disks. Usually, the first disk with index 0 must be selected. Then `list partition` determines the partitions. Once again, you must use `select` to select a partition for further processing—usually the last one. The `extend` command now enlarges it to the maximum size.

Chapter 32

QEMU and Kernel-Based Virtual Machine

Kernel-Based Virtual Machine (KVM) is a Linux-specific virtualization technology for the server and enterprise segment. It extends *Quick Emulator* (QEMU), which has been available for a while. Red Hat and SUSE fully rely on the combination of QEMU and KVM in their enterprise distributions, and all other popular Linux distributions also include QEMU/KVM packages.

This chapter first introduces the basics of KVM and then focuses on server virtualization using KVM: this allows multiple Linux virtual servers to run on one machine. In real life, this is often done to separate server functions as much as possible to maximize security. But practical reasons also often speak in favor of server virtualization. While one user needs special Apache modules for their website, another may want to use the latest MySQL version. If many special requests of this kind are fulfilled on *one* system, this quickly leads to unwanted side effects and instabilities.

Essentially, KVM is also suitable for desktop virtualization, but this aspect is of secondary importance here. If you still want to use KVM on desktop systems, you can try the *Boxes* GNOME program in addition to the Virtual Machine Manager presented in this chapter. As is typical of GNOME, it's very easy to use, but the program provides far too few configuration options even for simple applications.

KVM on the root server

By its nature, you can also use KVM on a root server. This enables you to set up several virtual machines on a real server, which look like your own servers to the outside world. There are two things you need to keep in mind:

- You need a *real* root server. Nowadays, many hosting and cloud providers offer virtual environments. This is cheap and flexible because you can easily add RAM or CPU cores later—but it makes virtualization impossible. (From a technical perspective, nested virtualization has been possible for several years, but in reality many problems and limitations still occur.)
- For most applications, both the KVM host and each guest require their own IP address. For numerous purposes, you need IPv4 addresses, which are rare and are therefore passed on by hosting providers at a correspondingly high price. In such cases, the routing between the KVM host and its guests is also relatively complicated.

Links

This chapter can only give an introduction to KVM. You can find information on this topic at the following pages:

- *https://www.linux-kvm.org*
- *https://libvirt.org*
- *https://help.ubuntu.com/community/KVM*

32.1 Basic Principles

KVM vs. QEMU

The QEMU program emulates various CPUs and elementary hardware components of a computer: its network card, a DVD drive for installation, and so on. QEMU is also able to emulate processors incompatible with the host CPU (ARM, Sparc, PowerPC, MIPS, etc.).

KVM is a kernel module that only unfolds its effect in combination with QEMU. KVM requires a CPU with hardware virtualization capabilities and turns the QEMU emulator into a hardware virtualization system. The elegance of KVM is that it does not perform typical hypervisor tasks itself but uses memory and process management functions of the Linux kernel to do so. The KVM functions are used via the `/dev/kvm` device file.

Hardware requirements

KVM works only if the host system processor supports virtualization features (Intel-VT or AMD-V). This is the case with most current processors. The exceptions include low-priced CPUs for cheap PCs or notebook computers. To determine if your CPU helps with hardware virtualization (Intel-VT or AMD-V), you should run the following `egrep` command. If the result is empty, your CPU does not support virtualization or the feature has been disabled in BIOS/EFI:

```
user$ egrep '^flags.*(vmx|svm)' /proc/cpuinfo
flags :... vmx ...
```

Even easier on Ubuntu systems, you can run the `kvm-ok` command from the `cpu-checker` package:

```
user$ kvm-ok
INFO: Your CPU supports KVM extensions
INFO: /dev/kvm exists
KVM acceleration can be used
```

For the remainder of this chapter, I will assume that your CPU is KVM-compatible. If it isn't, KVM *seems* to work as well. In fact, however, the virtual machines are only run through QEMU and thus without KVM support, and they then run much slower.

Enabling Virtualization Features in the BIOS/EFI

Surprisingly, the existing virtualization features of the CPU are often disabled by the BIOS or EFI. To fix this, when you boot the computer, open the BIOS/EFI dialogs and look for the relevant setting.

Required packages

To run KVM virtual machines and control them using the `libvirt tools`, you need to install the `qemu-system-x86` (`qemu-kvm` on openSUSE) and `virt-manager` packages, and also `libvirt` or `libvirt-client`, depending on your distribution. In the meantime, further development of KVM is taking place as part of the QEMU project. This is why many distributions no longer have separate packages for KVM:

```
root# apt install qemu-system-x86 virt-manag          (Debian, Ubuntu)
root# dnf install qemu-kvm virt-manager libvirt-client (Fedora, RHEL)
root# zypper install qemu-kvm virt-manager libvirt     (openSUSE)
```

Some distributions require you to explicitly start the `libvirtd` service:

```
root# systemctl enable --now libvirtd          (Fedora, openSUSE, RHEL)
```

Do Not Run an Open DNS Resolver on a root Server!

The `dnsmasq` program is installed together with the libvirt tools. This is a simple DHCP server and nameserver. The program is used to assign virtual machines their network parameters.

However, in some Debian and Ubuntu versions, `dnsmasq` is not only run by the libvirt tools, but started automatically, just like any installed network service. This means that an open nameserver is running on your computer.

At home or in private networks, this is not a big issue. However, if you run KVM, libvirt, and the like on a publicly accessible root server, an open nameserver is no good for you.

Fortunately, the security issue can be easily fixed. You just need to stop the program using `systemctl disable --now dnsmasq`. (The libvirt tools still use `dnsmasq`. Only the unconfigured operation of the program as a top-level service is stopped.)

Fortunately, this problem no longer occurs with current versions of Debian and Ubuntu: there, the libvirt tools only install `dnsmasq-base` with the program itself, but not `dnsmasq` with the systemd files to start as a background service.

Kernel modules

KVM provides its functions in three kernel modules: the basic functions are located in the `kvm` module, the Intel-VT-specific functions in `kvm-intel`, and the AMD-V-specific functions in `kvm-amd`. To use KVM, you must load the KVM module that matches your hardware. The `kvm` module is loaded at the same time. In most distributions, the init system takes care of this. If that does not work, you should intervene manually:

```
root# modprobe kvm-intel  (for Intel-VT processors)
root# modprobe kvm-amd    (for AMD-V processors)
```

Conflicts between Multiple Virtualization Systems

Note that only one program can use the virtualization functions of the CPU at a time. That is why it is impossible to use VirtualBox and KVM at the same time.

KVM command

In the past, KVM virtual machines were run using the `kvm` or `qemu-kvm` command. Today, the command is `qemu-system-x86_64` for almost all distributions. Only RHEL and compatible systems use `/usr/libexec/qemu-kvm` instead.

After starting the command, the virtual machine will be displayed in a window or must be controlled by a VNC client. In general, however, a direct use of the command is rarely recommended. You need to set the virtual machine hardware components through countless options, and this makes the application confusing and prone to errors.

libvirt tools

The `libvirt` tools simplify the administration of virtual machines:

- The Virtual Machine Manager (program or package name `virt-manager`) helps with a graphical user interface to set up and run virtual machines.
- If you prefer to work in the terminal, you can use the `virsh` shell to create, start, and stop virtual machines and perform other admin tasks.
- In addition, there are various commands for special tasks: `virt-clone` copies a virtual machine; `virt-top` returns a list of all virtual machines, including RAM and CPU usage; and so on, similar to `top`.

Distributions

All major distributions provide QEMU/KVM and `libvirt` packages. Because the development of KVM is driven to a very high degree by Red Hat, RHEL or RHEL clones are particularly well-suited for KVM deployment. If you want to test the very latest KVM features, Fedora is the ideal playground. However, this doesn't mean that other distributions are not equally suitable. Personally, I have been running KVM hosts under various versions of Ubuntu for over a decade.

32.1.1 Internal Workings of libvirt

libvirt is an interface for managing virtual machines and their associated virtual network and disk devices. A prerequisite for using the libvirt tools is that the host system must be running the libvirtd daemon. This program is started by the init system when the host computer is booted.

The virtual machines are controlled by the virsh shell, by the Virtual Machine Manager, or by other libvirt commands. Each of these programs must connect to the libvirt daemon beforehand. The libvirt daemon also allows network connections that are usually tunneled through SSH.

In addition to KVM, the libvirt tools can also control the Xen virtualization system. In this chapter, however, I refer exclusively to KVM.

System level versus user level

KVM machines can run on two levels via libvirt:

- **User level (qemu:///session)**
  This variant is primarily intended for desktop virtualization and grants the virtual machines less access to the host computer's hardware. Internally, the first time a libvirt tool is called at the user level, a separate libvirtd process is started that has only the permissions of the current user. Thus, user-level KVM machines minimize the security risks posed by virtualization.
- **System level (qemu:///system)**
  System-level virtual machines are better suited for server virtualization because they can directly access hardware components of the host computer and because there are more options for integrating the virtual machines into the network. The libvirt processes communicate with the libvirtd daemon, which runs with root privileges.

Fedora and RHEL

There are strong configuration differences between distributions in the communication between the libvirt tools and the libvirtd daemon. It's quite simple with Fedora, RHEL, and the like: if you want to communicate with libvirtd at the system level, you need root privileges. The Virtual Machine Manager can be started with user rights, but the program expects the root password to be specified immediately after startup.

Note that the libvirt tools are executed with root privileges, but not the actual virtualization command! Rather, the libvirt tools launch the qemu-kvm command under the qemu user account. You have to pay attention to this subtlety, especially to ensure you're setting the access rights for image or ISO files correctly!

Ubuntu

Also on Ubuntu, libvirt tools running with root privileges communicate with libvirtd at the system level. But even libvirt commands executed

only with user privileges are allowed to communicate with `libvirtd` at the system level, so long as the user belongs to the `libvirtd` group! Strictly speaking, what matters is whether the user is allowed to access the `/var/run/libvirt/libvirt-sock` file. This file belongs to `root` and the `libvirt` group.

The association with the `libvirt` group is automatically established for the user performing the installation when the `libvirt-bin` package is installed. Additional users can be added to the `libvirt` group with the following command:

```
root# usermod -a -G libvirt <username>
```

Note that the new group assignment will not take effect until you log out and log in again!

Debian and openSUSE

In Debian and openSUSE, all users have read and write permissions for `/var/run/libvirt/libvirt-sock`. A group assignment is therefore not necessary.

Configuration

The configuration files of the `libvirt` system are located in the `/etc/libvirt` directory. The `qemu.conf` file contained in this directory is particularly interesting: it specifies various basic settings for the KVM command. Among other things, the file controls the default settings of the virtual machine's VNC or Spice server. The IP address 127.0.0.1 is used by default. Thus, only local connections are allowed, although forwarding via SSH is possible through port forwarding.

In addition, the properties of each virtual machine are recorded in an XML file in the `/etc/libvirt/qemu` directory. Most settings are understandable without further explanation and correspond directly with corresponding options of QEMU/KVM. The format of `libvirt` XML files is documented in detail on the following page: *https://libvirt.org/format.html*.

Always Change the Description of Virtual Machines by Using "virsh edit"!

You should not modify the XML files containing the key data of a virtual machine directly in an editor, as otherwise another `libvirt` tool may overwrite your changes. Instead, you should use the `virsh edit` command!

Image files

If you use disk images to map the virtual data carriers when setting up virtual machines, these are saved in the `/var/lib/libvirt/images` directory by default. If you want to store disk images in another directory or use logical volumes, disk partitions, or network devices to store the disks, you need to set up a so-called storage pool beforehand. The easiest way to do this is in

Virtual Machine Manager. Alternatively, you can run the appropriate `pool` commands within `virsh`.

32.1.2 Behavior when Rebooting the Host System

What happens to virtual machines when you shut down the host system? Many distributions back up the memory contents of all virtual machines that are currently being executed by `libvirtd` at the system level. Internally, the `save virsh` command is used. The next time the computer is started, the state of the virtual machine is automatically restored; that is, the virtual machine continues to run as if nothing had happened in the meantime.

When backing up or restoring multiple virtual machines, all of their RAM must be saved to or read from disk. This requires sufficient free space in the `/var/lib/libvirt/qemu/save` directory and of course takes some time.

However, the detailed implementation varies from distribution to distribution:

- **Fedora and RHEL**
  The `/usr/libexec/libvirt-guests.sh` script, which is called by the `libvirt-guests` systemd service, is responsible for this mechanism. You can store configuration parameters in `/etc/sysconfig/libvirt-guests`. This file does not exist by default. The allowed parameters can be found in `libvirt-guests.sh`.
- **Debian and Ubuntu**
  Here, the `/usr/lib/libvirt/libvirt-guests.sh` script is executed. It takes into account settings from `/etc/default/libvirt-guests`.
- **openSUSE**
  Current SUSE distributions also use the same mechanism as RHEL. However, the corresponding script is located in `/usr/lib64/libvirt/libvirt-guests.sh`.

On some distributions, the storage service is not automatically active, but must be explicitly enabled using `systemctl`:

```
root# systemctl enable --now libvirt-guests   (Fedora, openSUSE, RHEL)
```

32.1.3 Virtual Hardware

When setting up a new virtual machine, you have a lot of choices: disk images in RAW or QCOW2 format, IDE or virtio hard disk adapters, the graphics system based on SDL, VNC, or Spice, and so on. This section summarizes the most important information in this area.

Virtio drivers and paravirtualization

Basically, KVM performs a complete virtualization. Thus, the guest system running in the virtual machine does not require any special drivers.

The guest system can be run even more efficiently if the optional virtio drivers are used for the communication between KVM and the virtual machine. Technically speaking, this is referred to as *paravirtualization*; that is, the guest system helps with the virtualization.

For Linux guests, virtio drivers are available by default to accelerate disk, memory, and network access. So it's just a matter of selecting the appropriate virtio components when setting up the virtual machine.

The matter is a little more difficult if you want to run Windows in a KVM machine. In this case, you first need to set up the virtual machine with traditional hardware components—for example, with a virtual SATA interface. After installing Windows, you must install the virtio drivers, and only then can you activate the virtio components by modifying the virtual machine afterward.

For Windows 10, this approach has always worked well in my experiments, but I have reached my limits with Windows 11—or perhaps the limits of QEMU/KVM or Windows. After changing the drivers, the Windows boot process failed. My only Windows 11 installation to date uses a virtual SATA hard disk.

Virtual disks

To offer a virtual disk to a guest, an image file is often used on the KVM host. QEMU/KVM support two image formats:

- **RAW format**
  With the RAW format, the blocks of the virtual hard disk are simply mapped 1:1. If the file system of the host computer supports so-called sparse files, blocks containing only zeros are not physically stored. This is how it works with the `ext`, `xfs`, and `btrfs` file systems, among others, and this initially saves a lot of space. The RAW format is the simplest and fastest image format for virtual machines.
- **QCOW2 format**
  QCOW2 stands for *Qemu Copy-on-Write, Version 2*. This format provides numerous additional features compared to RAW. The data blocks are only reserved when required, without requiring a file system that is compatible with sparse files. In addition, the virtual file system can be compressed and encrypted. Finally, QCOW2 images provide the ability to manage snapshots. QCOW2 images are slower than RAW images, but the speed disadvantage is not as great as it used to be.

QCOW2 has been the default file system of Virtual Machine Manager for several years. The development of a third format (QED) was discontinued.

Instead of image files, you can also use disk partitions, logical volumes, or iSCSI devices as virtual disks. These variants provide administrative advantages in large virtualization systems but do not offer significantly higher speed compared to RAW images.

Network connection

To connect the virtual machine to the local network or the internet, you must equip it with a network adapter. For Linux guests, the virtio driver is the first choice. For Windows guests, QEMU simulates an Intel E1000 network adapter.

The second question is how to connect the adapter to your network:

- **NAT**
  By default, the KVM command or the `libvirt` tools opt for the Network Address Translation (NAT) variant. This forwards the host's internet access to the guest. However, the guest is not visible on the local network or on the internet.

  There is an important difference in the NAT implementation here compared to VirtualBox: SSH connections from the host to the virtual machine are possible without any problem.
- **Network bridges**
  For server deployment, you must connect the virtual machine to the network or internet through a network bridge or routing. This requires a special network configuration of the KVM host (Section 32.4).
- **MacVTap**
  Here, a virtual network adapter is directly connected to a physical adapter.

Illustration

At least during the installation, you need to see the virtual machine output. This means the guest needs a separate graphics system. This is done by emulating a VGA-compatible graphics card, whose outputs are then displayed in a window via VNC or Spice. For 2D use, this works well even with high resolution.

VNC and Spice are network-compatible. The slightly higher speed and the fact that virtual machine audio output can also be forwarded to the local Spice client via the network speak in favor of the Spice architecture.

32.2 Virtual Machine Manager

Virtual Machine Manager (program and package name `virt-manager`) is the best way to get familiar with KVM. The program has a lot to offer even for experts and is absolutely sufficient for many virtualization tasks.

To use the program, you must connect to the `libvirt` daemon. In most distributions, the connection to the existing QEMU/KVM item is established automatically. On Fedora, openSUSE, RHEL, and the like, you need to enter the `root` password when starting Virtual Machine Manager.

On Ubuntu, the connection works only if the user is part of the `libvirtd` group. If that is not the case, you must run `sudo adduser loginname libvirtd` and log out and log in again.

Soon to Be Obsolete?

With the introduction of RHEL 8, Red Hat has given the `virt-manager` package the *deprecated* attribute. Red Hat now wants you to set up virtual machines in Cockpit using the `cockpit-machines` package. Although this works, `cockpit-machines` can not keep up with the functionality of Virtual Machine Manager. The deprecated note is currently understood to mean that Red Hat doesn't want to continue developing `virt-manager` and sees the future in Cockpit. Let's wait and see if users will see it that way.

Another alternative to Virtual Machine Manager is Proxmox Virtual Environment (see *https://www.proxmox.com*). This is a separate distribution based on Debian. It includes QEMU, KVM, and libvirt as well as a web interface for virtual machine management. According to the freemium model, the basic version is freely available. However, additional functions, updates, and support are subject to a charge. Proxmox is not very easy to test because the entire disk is erased during the installation. Parallel operation with other distributions is not intended.

Running virtual machines

The main window of Virtual Machine Manager contains a list of all `libvirt` connections (see Figure 32.1). By default, this list consists of only one entry, **QEMU/KVM**, for the local virtualization system. However, you can specify the key data of additional KVM hosts using **File • Add Connection**.

Double-clicking an entry in this list establishes the connection to the KVM host and then displays all virtual machines of this host. For each virtual machine, an icon shows whether the machine is shut down, running, or paused.

To start a virtual machine, right-click its entry and execute **Run**. Double-clicking the entry opens a new window that displays the state of the virtual machine in two views:

- The *console view* shows the graphics system of the virtual machine. There you can see the virtual machine's output and you can input data using the keyboard and mouse. For virtual machines running in text mode, clicking in the virtual machine captures the mouse cursor, as it were. Ctrl+Alt releases the cursor again.

- The *detail view* shows the key data of the virtual machine. There you can change the hardware configuration of the virtual machine. However, most changes can be made only if the virtual machine has been shut down before that.

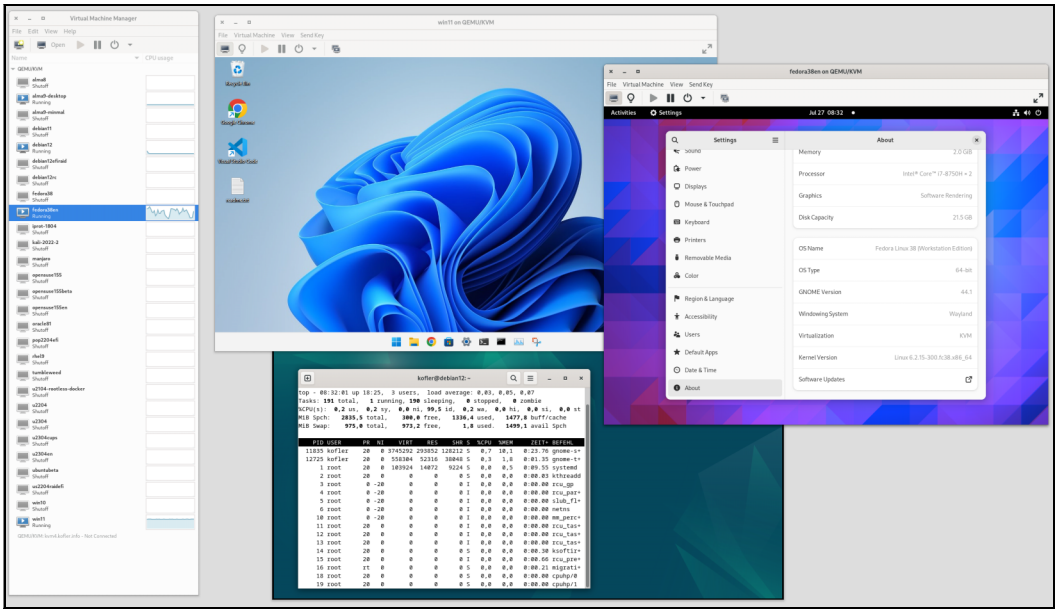


Figure 32.1 Virtual Machine Manager with Various Virtual Machines

To switch between the two views, you need to run **View • Console** or **View • Details** or click the corresponding buttons in the toolbar.

Administration within the network via SSH

You can also use Virtual Machine Manager to administrate an external KVM host that can be accessed via SSH. To do this, you first need to define a connection to this server using **File • Add Connection**, where you select **SSH** as the connection method (see Figure 32.2). When establishing the connection, you must also enter the corresponding login password.

If the external KVM host is running RHEL or Fedora, the `libvirt` administration requires `root` privileges and thus a `root` login via SSH. For security reasons, however, SSH servers are often configured in such a way that direct `root` login is impossible. A possible compromise is to configure the SSH server in such a way that a `root` login is only accepted if authenticated by a key, but not by password.

For the administration of external KVM hosts, it is generally recommended to use SSH keys for authentication. To do this, you want to copy the public part of your key to the remote computer before establishing the first connection via `ssh-copy-id` (see Chapter 23, Section 23.4).

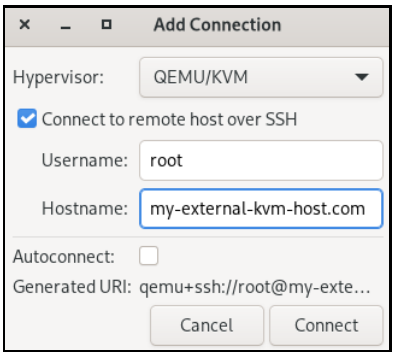


Figure 32.2 Connection Setup via SSH

32.2.1 Setting Up a New Virtual Machine

Setting up a new virtual machine starts with the **Create a new virtual machine** button. A five-step wizard helps you to set the key data:

- In the first step, you specify on which KVM host the virtual machine is supposed to be installed. This selection option is relevant if you use Virtual Machine Manager to administrate not only one local KVM host, but multiple virtualization machines.

  There, you also specify the installation source. A Linux installation is usually an ISO file. However, it is also possible to read the installation data from the DVD drive of the host computer. The **Import Existing Disk Image** option creates a copy of an existing virtual machine image file.
- If you selected an ISO image or a CD/DVD as the installation source in the first step, you can specify the filename of an ISO file or the CD/DVD drive in the second step. The **Browse** button leads to a dialog that initially only displays the **storage Pools** known to Virtual Machine Manager. To select an ISO file directly, you must click the **Browse Local** button in this dialog.

  Sometimes the wizard manages to get the type of the operating system and its version from the installation media—**Linux** and **Fedora**, for example. If this does not succeed, you will have to provide this data yourself. These settings are necessary for the wizard to set up the optimal virtual hardware components for the guest system.
- In the third step, you must specify how much memory (RAM) and how many CPU cores you want to allocate to the virtual machine.
- In the fourth step, you will set up the virtual hard disk. Usually, you will use the already preselected **Create Disk Image for the virtual machine** option. By default, the new image file will be created in the `/var/lib/libvirt/images` directory. Do not select a virtual hard disk size that is too small as subsequent enlargement is only possible with great effort.

Alternatively, the **Select and Create Custom Storage** option and the associated **Manage** button will take you to another dialog for managing all libvirt disks. There you can set up multiple storage pools. Such a pool can be, for example, any directory in the file system or an LVM system. Within the selected pool, you can then create a new disk and select it for installation.

- In the fifth step, you assign a name to the virtual machine and choose between different network options. By default, Virtual Machine Manager uses the NAT method. This method assigns the virtual machines to the private network 192.168.122.0/24. Thus, the virtual machine can use the host computer's internet access but cannot connect to other computers on your local network.

  The **Done** button finishes the configuration and starts the new virtual machine immediately. If you do not want this, you should enable the **Customize Configuration before Install** option in the final dialog sheet of the wizard. This will take you to a dialog that summarizes the virtual machine hardware components after you finish the wizard (see Figure 32.3).

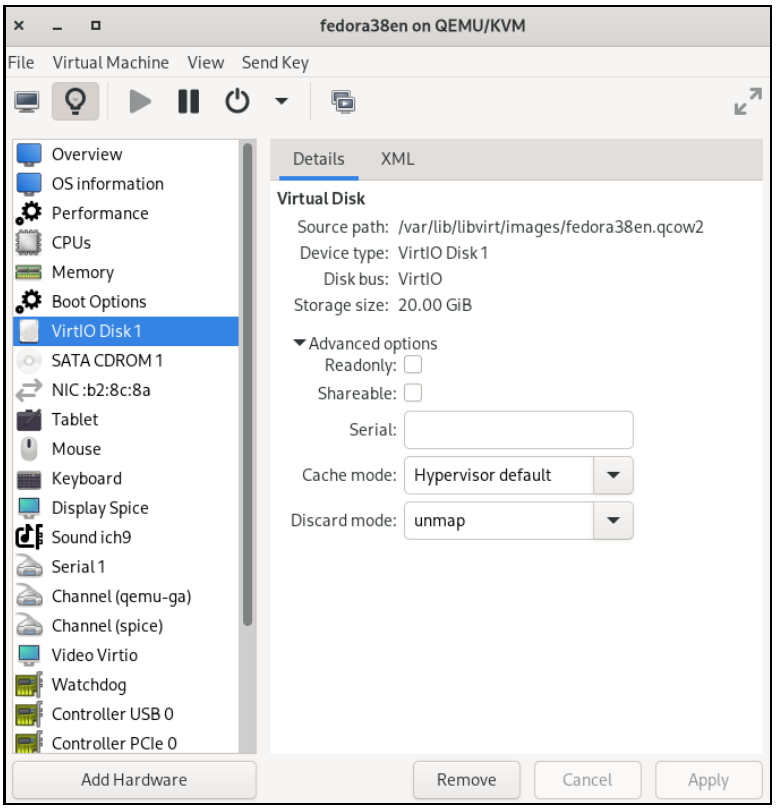


Figure 32.3 Hardware Management in Virtual Machine Manager

The virtual machine will start as soon as you finish the configuration. You will see the virtual machine output in a new Virtual Machine Manager window. Behind the scenes, this window acts as a VNC or Spice client.

BIOS or EFI By default, virtual machines are equipped with BIOS. If you want to experiment with EFI in your virtual machines, you must first install an EFI package. On Debian and Ubuntu, the required files are in the `ovmf` package, on Fedora in `edk2-ovmf`. For Virtual Machine Manager to take the new package into account, you must restart `libvirtd` using `systemctl restart libvirtd`.

When setting up new virtual machines, you must then make sure that you click the **Customize Configuration before Install** option in the final step of the wizard. After that, you can choose between the BIOS and three EFI variants (with or without Secure Boot) in the **Overview** dialog sheet under **Firmware** in the virtual machine's detailed settings.

Clipboard For the exchange of text via the clipboard between the host and the virtual machines to work, the `spice-vdagent` package must be installed in the virtual machines.

32.2.2 Stopping Virtual Machines

The virtual machines run completely independently of Virtual Machine Manager! So you can close the Virtual Machine Manager windows and open them again later, while the virtual machines will continue to run in the meantime. You can even log out and log in again without affecting the execution of virtual machines.

There are many ways to stop a virtual machine:

- **Shutdown • Reboot** sends an appropriate ACPI event to the virtual machine. When the virtual machine processes ACPI events, it initiates a shutdown and then a reboot.
- **Shutdown • Shutdown** initiates a shutdown via ACPI.
- **Shutdown • Force Reset** or • **Force Off** stops the virtual machine execution immediately—just as if you would unplug the power cable from a real computer. Needless to say, you should try to avoid this type of shutdown, as it can result in data loss. The difference between the two commands is that the virtual machine is restarted after the **Reset**.
- **Shutdown • Save** saves the contents of the machine's virtual RAM to a file and then stops execution. When the virtual machine is restarted later, it will be in exactly the same state as when it was shut down.
- Of course, you can also shut down or restart the virtual machine itself—for example, by running the `shutdown now`, `reboot`, or `halt -p` commands within the machine.

32.2.3 Windows Installation

Windows 11 has higher hardware requirements than previous versions. The virtualization system must support EFI including Secure Boot and TPM for the operating system to run.

Current versions of QEMU/KVM and Virtual Machine Manager support Windows 11 out of the box. In my tests with Ubuntu 23.04 as the host system, Virtual Machine Manager independently detected the Windows 11 operating system after selecting the ISO file.

If you use an older distribution, on the other hand, some manual work is required:

- You need to install UEFI firmware files and a software implementation of TPM. The required packages are named `ovmf` or `edk2-ovmf`, `swtpm`, and `swtpm-tools`. Depending on the distribution, you may need to activate additional package sources, such as Stefan Berger's PPA (*https://launchpad.net/stefanberger*) or EPEL.
- UEFI must be set as firmware (no BIOS). You can find the option in the **Overview** dialog sheet. Note that the firmware cannot be changed once the installation begins.
- Finally, the virtual machine must contain the TPM hardware component. If the virtual machine setup assistant does not take care of it itself, you can use the **Add HArdware** button.

32.3 libvirt Commands

Now that I have presented Virtual Machine Manager as the most important representative of the `libvirt` tools in the previous section, here are some commands whose use does not require a graphical user interface.

32.3.1 virsh

You can use the `virsh` command to start the `libvirt` shell. In it, you can execute commands to manage all virtual machines known to `libvirt`:

```
root# virsh
virsh# list --all
virsh # list --all
 Id   Name                    State
----------------------------------------
 1    alma9-desktop           running
 2    debian12                running
```

```
 4    win11                    running
 -    alma9-minmal             shut off
...
virsh# start fedora
Domain fedora started
virsh# exit
```

Using `virsh`, you can use the following commands to start, temporarily suspend and resume, shut down in an orderly fashion (ACPI shutdown), shut down immediately, save and resume, or delete virtual machines from the list of `libvirt` definitions: `start`, `suspend`/`resume`, `shutdown`, `destroy`, `save`/`restore`, or `undefine`.

`edit` allows you to edit the XML file with the virtual machine description directly in an editor. If the `EDITOR` environment variable is not initialized, `virsh edit` asks which of the installed editors you want to use the first time. `virsh` then remembers your answer.

For all `virsh` commands listed thus far, you must specify the virtual machine name, its UUID number, or, for running virtual machines, the ID number. The UUID number derives from the XML definition file. Information about the ID numbers of running virtual machines is provided by the `list virsh` command.

Establishing the connection

Provided `virsh` detects that the machine is a KVM host, it connects to the `libvirt` daemon running at the system level if possible. When `virsh` is run with user privileges only, a system-level connection on Ubuntu requires membership in the `libvirtd` group. On Fedora and RHEL, you must run the command as `root` if you want to work at the system level.

It's possible to change the connection within the `virsh` shell using the `connect` command. `qemu:///session` denotes a user-level connection and `qemu:///system` a system-level connection:

```
virsh# connect qemu:///system
```

You can also connect to the `libvirtd` daemon on another machine via SSH. If the KVM host is a RHEL or Fedora machine, you must specify `root` as the username because only `root` is allowed to connect to the `libvirt` system daemon on these systems. Note that `qemu+ssh:` is followed by only two slashes, not three! If a `root` login with password via SSH is not possible on the KVM host for security reasons, you must set up your public SSH key on the KVM host before the first connection is established:

```
virsh# connect qemu qemu+ssh://user@hostname/system
user@hostname's password: *******
```

Instead of using virsh interactively, you can run a single virsh command using the format virsh command:

```
root# virsh  list --all
...
```

If you do not want to use the connection provided by virsh by default, you need to specify the connection string with the -c option:

```
root# virsh -c qemu:///session list --all
...
```

Selected virsh commands

Finally, I want to present some selected virsh commands. man virsh documents at least a hundred more commands. Within the shell, you get a detailed description of the respective command via help name:

- connect qemu:///session establishes an ordinary user connection to libvirtd. This way you can manage your own virtual machines.
- connect qemu:///system establishes a root connection to libvirtd. This is only necessary if global KVM options or virtual networks are to be changed.
- list [--inactive or --all] lists all running virtual machines. If you want to list only the currently inactive machines or both inactive and running machines, you need to specify the --inactive or --all options.
- start <vmname> starts the specified virtual machine. If you want to communicate with the machine in graphics mode, you can use either the virt-viewer program or a VNC client. The connection data is determined by the vncdisplay virsh command.
- suspend/resume <vmname> temporarily stops or resumes the specified virtual machine. The stopped virtual machine will continue to consume RAM! So only the virtual CPU is stopped.
- shutdown/reboot <vmname> shuts down or reboots the virtual machine. The virtual machine receives a shutdown signal via ACPI. However, it's up to the virtual machine to decide whether it will respond. If this is not the case with older distributions, installing the acpid package in the virtual machine usually helps.
- save <vmname> <filename> saves the virtual machine state to a file and then stops the machine from running.
- restore <filename> reactivates the previously saved virtual machine. The state file can be deleted afterward.

- `destroy <vmname>` terminates the virtual machine immediately. This is like unplugging the power cord on your computer—with the same consequences.
- `undefine <vmname>` deletes the XML file that describes the virtual machine. The image file with the virtual hard disk will be preserved. If you want to delete the image file(s) as well, you need to use the `--remove-all-storage` option.
- `autostart [--disable] <vmname>` specifies that the virtual machine should start automatically during the host boot process. By using the `--disable` option, you can deactivate the automatic start again. Automatic startup works only for machines that are set up at the system level. On the other hand, at the session level, `autostart` machines are not started until `virsh` is used to connect to `libvirtd` for the first time.
- `console <vmname>` allows you to operate the specified virtual machine directly in the console. This assumes that a `getty` process for the serial `/dev/ttyS0` port is running in the virtual machine. To terminate the connection, you need to press Ctrl + D.
- `ttyconsole <vmname>` specifies which device of the host computer is used to access the serial port of the guest system (e.g., `/dev/pts/5`). You can then run `socat - /dev/pts/5` in a terminal window and communicate with the virtual machine (much like the `console` command described previously). Before that, the `socat` package must usually be installed.
- `vncdisplay <vmname>` returns the IP address (empty for `localhost`) and port number for the VNC display of the virtual machine. You can then launch any VNC client (such as Vinagre) to interact with the virtual machine. On the KVM host, you can also run `virt-viewer vmname` instead. For security reasons, VNC access works only from `localhost`.

  `vncdisplay` does not return a result if the virtual machine does not share its graphics system via VNC at all but uses the more modern Spice system instead. In this case, you can operate the virtual machine using the `virt-viewer` program. You can directly pass the virtual machine name to this program.

  However, should the need arise to determine the Spice port number, it will become difficult. `virsh` lacks a command to find the Spice port of a virtual machine, and that's similar to `vncdisplay`. The following command extracts the port number from the process list:

  ```
  ps aux | grep vm_name | grep -oP "(?<=-spice port=).*?(?=,)"
  ```

- `edit <vmname>` loads the XML file describing the virtual machine into the editor you set in the `$EDITOR` environment variable.

32.3.2 virt-clone

Setting up a new virtual machine usually takes quite some time. If you want a virtual machine that has essentially the same key data as an existing virtual machine, it's much faster to simply copy or "clone" it. The `virt-clone` command can help with this: by default, it creates a new XML definition file, copies the virtual disk image file, and assigns a new random MAC address to the network adapter. The remaining hardware components remain unchanged. The virtual machine must be shut down before copying.

The following command copies an Ubuntu server installation. The new virtual machine is named `userver6`, and the new image is saved as the `/var/lib/libvirt/images/userver6.img` file:

```
root# virt-clone --original userver5 --name userver6 \
                 --file /var/lib/libvirt/images/userver6.img
```

SELinux

Be sure to create the new image file in a libvirt storage pool. Otherwise, the SELinux rules on RHEL/Fedora will prevent the virtual machine from running.

After copying, you need to make various adjustments in the virtual machine. For example, you need to change the network configuration so that there are no IP address conflicts. Depending on the configuration, it may be necessary to update the `/etc/hosts` and `/etc/hostname` files as well. If there was an SSH server in the original virtual machine, you must make sure to create a new SSH key in the virtual machine:

```
root# systemctl stop sshd                    (Debian/Ubuntu)
root# rm /etc/ssh/ssh_host_*
root# dpkg-reconfigure openssh-server

root# systemctl stop sshd                    (Fedora/RHEL)
root# rm /etc/ssh/ssh_host_*
root# systemctl start sshd
```

32.3.3 virt-sysprep

Instead of modifying the virtual machines manually, you can use the `virt-sysprep` command for this purpose. The command first resets various data (SSH keys, entries in the cron spool directory, etc.) and then performs the

operations specified by options. The following command first resets the virtual machine named `alma` and then sets a new host name:

```
root# virt-sysprep --hostname mynewhostname -d alma
Examining the guest ...

Performing "bash-history" ...

Performing "cron-spool" ...

Performing "dhcp-client-state" ...

Performing "logfiles" ...
...

Setting the hostname: testhostname
```

You can prevent the reset operations performed by default by passing to the `--operations` option a list of actions to be performed. The following command changes only the host name, but does not change the virtual machine in any other way:

```
root# virt-sysprep --hostname mynewhostname --operations customize \
        -d alma
Setting the hostname: testhostname
```

By using `virt-sysprep`, you can also add users; change passwords; install packages; add, change, or delete files; and more. The numerous options and operations are described in the `man` page.

Many operations work independently of the distribution in the virtual machine. `virt-sysprep`, however, is developed by Red Hat and is therefore particularly well-suited if the virtual machine contains RHEL or a distribution compatible with it.

32.3.4 virt-viewer

`virt-viewer` from the package of the same name is a VNC and Spice client for displaying the screen contents as well as communicating with a virtual machine. `virt-viewer vmname` establishes the connection to a running virtual machine. This assumes that the virtual machine uses VNC or Spice:

```
root# virt-viewer <vmname>
```

If you run `virt-viewer` without `root` privileges but want to connect to a virtual machine running at the system level, you must specify the connection string with the `-c` option:

```
user$ virt-viewer -c qemu:///system <vmname>
```

Provided that the virtual machine uses VNC to share the virtual graphics system, you can use any other VNC client instead of `virt-viewer`. The only difference is that you must first use the `virsh` command `vncdisplay` to determine the connection data.

Spice clients for Windows and macOS can be found at *https://www.spice-space.org/download.html*.

32.3.5 virt-top

The `virt-top` command from the package of the same name provides a listing of all virtual machines, similar to `top`. For each virtual machine, its memory and CPU requirements as well as various other parameters are displayed:

```
user$ virt-top
virt-top 07:29:53 - x86_64 12/12CPU 2200MHz 31617MB
31 domains, 2 active, 2 running, 0 sleeping, 0 paused, 29 inactive
D:0 O:0 X:0
CPU: 0.1%  Mem: 3072 MB (3072 MB by guests)

   ID S RDRQ WRRQ RXBY TXBY %CPU %MEM    TIME   NAME
    1 R    0    0   52    0  0.1  6.0   8:46.14 ubuntu
    2 R    0    0    0    0  0.0  3.0   0:21.59 alma
    -                                           (fedora)
   ...
```

32.4 Integrating Virtual Machines into the LAN (Network Bridge)

By default, `libvirt` uses so-called user-mode networking. The virtual machines are assigned an IP address in the 192.168.122.* range via DHCP. On the host computer, you need to set up the IP address 192.168.122.1 as the gateway to the internet. This way, the guests have internet access and can also communicate with the host via its address, 192.168.122.1. Apart from that, KVM guests cannot access network services on the local network, and vice versa, while computers on the LAN cannot communicate with KVM guests either.

To provide server services to the local network on KVM guests, you need a virtual network bridge that connects the virtual network adapters of the KVM machines to the physical network adapter of the host computer. What can be achieved in VirtualBox quite simply by selecting a different network type is unfortunately associated with manual work in `libvirt` or Virtual Machine Manager.

32.4.1 Configuring the Network Bridge on the Host Computer

To build the bridge, you can use the tools from the `bridge-utils` package. However, as usual, the configuration details vary from distribution to distribution. I want to present the procedures for Fedora/RHEL, Ubuntu, and Debian here. For all variants, you should make sure that you replace the name of the network interface (`enp0s1` in the following examples) with the interface name that is relevant for you!

Fedora and RHEL

On Fedora and RHEL, NetworkManager is responsible for bridge building. If you use a desktop system, it's best to start the `nm-connection-editor` graphical user interface. Using this, you first need to set up a new bridge and add a connection (**Bridged Connection**) to the relevant Ethernet interface in the **Bridge** dialog sheet. Back on the main page of the program, you then want to remove the Ethernet interface. The network connection should thus take place automatically via the bridge. (In my tests, this sometimes required a reboot—quite untypical for Linux.)

If you run QEMU/KVM on a root server, there will probably be no desktop system, and the `nm-connection-editor` will accordingly not be available. You can help yourself by using the `nmtui` text-based configuration tool (**Edit • Bridge • Create**), or you can create the bridge using `nmcli` commands as in the following example:

```
root# nmcli connection add type bridge ifname br0 stp no
root# nmcli connection add type bridge-slave ifname enp0s1 master br0
root# nmcli connection show --active (returns UUIDs from all interfaces)
root# nmcli connection down 659b...  (UUID of enp0s1)
root# nmcli connection up bridge-slave-enp0s1
```

You will find helpful information on building bridges with the NetworkManager on the following two pages:

- *https://www.xmodulo.com/configure-linux-bridge-network-manager-ubuntu.html*
- *https://linuxconfig.org/how-to-use-bridged-networking-with-libvirt-and-kvm*

Theoretically, it's possible to set up a network bridge to a Wi-Fi interface in a similar way. However, that's not recommended (see *https://pve.proxmox.com/wiki/WLAN*).

Ubuntu

For Ubuntu, the network configuration is based on `netplan` (see Chapter 15, Section 15.4). To build a bridge, use the following configuration file as a guide:

```
# file /etc/netplan/02-ethernet-bridge.yaml
network:
  version: 2
  renderer: networkd
  ethernets:
    enp0s1:
      dhcp4: no
  bridges:
    br0:
      dhcp4: yes
      interfaces:
        - enp0s1
```

In this case, the `enp0s1` interface is connected to the `br0` bridge. The bridge obtains its IP configuration from the DHCP server on the local network. (Alternatively, you can of course configure the bridge statically.) To test the syntax of the new configuration file, you must run `sudo netplan apply`. However, the bridge does not become active until after a reboot.

Debian

On Debian, bridge building is done in the `/etc/network/interfaces` file. The existing lines for manual configuration of the interface to the LAN (in this example, `enp0s1`) must be changed so that this interface can then be controlled manually. For this purpose, the corresponding configuration settings now move to the description of the `br0` interface (or whatever you call the bridge).

In this example, 10.0.0.138 is the gateway and the DHCP server of the local network. The bridge itself is assigned the IP address previously held by the host computer (10.0.0.120). Don't forget that `/etc/resolv.conf` must contain the address of the name server:

```
# file /etc/interfaces/network (Ubuntu)
# loopback network interface (unchanged)
auto lo
iface lo inet loopback

# interface to LAN (manual)
auto enp0s1
```

```
iface enp0s1 inet manual

# bridge to enp0s1
auto br0
iface br0 inet static
       address 10.0.0.120
       network 10.0.0.0
       netmask 255.255.255.0
       broadcast 10.0.0.255
       gateway 10.0.0.138
       bridge_ports enp0s1
```

You can use `ifdown/ifup` to restart the affected network interfaces. The `br0` bridge now has the IP address 10.0.0.120 and transmits the IP packets to the physical network adapter, `enp0s1`. If the bridge is supposed to obtain its IP address via DHCP, the configuration of the `br0` interface becomes easier:

```
# file /etc/interfaces/network
# loopback network interface (unchanged)
auto lo
iface lo inet loopback
# interface to LAN (manual)
auto enp0s1
iface enp0s1 inet manual
# bridge to enp0s1
auto br0
iface br0 inet dhcp
       bridge_ports enp0s1
```

32.4.2 IP Forwarding

Regardless of the distribution, you must enable IP forwarding if you have not already done so. To do this, you need to enter two lines at the end of `/etc/sysctl.conf` and then run `sysctl -p`:

```
# at the end of /etc/sysctl.conf
...
# forwarding for IPv4 and IPv6
net.ipv4.ip_forward=1
net.ipv6.conf.all.forwarding=1
```

32.4.3 Network Configuration on a Root Server

The configuration of the network becomes a little more complicated if you want to make several guests publicly accessible on the internet on a root

server. The task then is to correctly route all traffic for multiple IP addresses through the host to the guests. The specific procedure depends strongly on the technical conditions of the hosting provider.

32.4.4 Configuring the Virtual Machine

When setting up the virtual machine, you must specify the name of the bridge as the network source, not the Ethernet interface on which the bridge is based. Virtual machines can use network bridges only when they run at the system level. Here's why: The network communication between the host computer and the KVM guest is handled by so-called TUN/TAP devices. These are network interfaces simulated by the kernel that are set up each time the virtual machine gets started. Fortunately, the `libvirt` tools take care of all the details, but can only do their job with `root` privileges.

32.4.5 MAC Trouble

The use of a network bridge has a serious disadvantage: it leaves the MAC addresses (usually 52:54:00:nn:nn:nn:nn) of the virtual machines contained in the IP packets unchanged. Hosting providers don't like that at all: outgoing packets must have only known MAC addresses assigned to the leased servers. If you do not adhere to this rule, the root server may be locked.

But there are several ways out. One variant is to perform a routing setup instead of a network bridge. Alternatively, you can request MAC addresses from the provider and enter them into the network configuration of your virtual machines (the easiest way is to use `virsh edit`). However, this only makes sense if you run only a few virtual machines. The last time I set up a larger KVM host, I came across a third way that can be done with minimal effort. You can use the `ebtables` command to overwrite all MAC addresses in forwarded packets with the MAC address of your root server's network adapter:

```
#!/bin/bash
# file /etc/rc.local on the host system
ebtables -t nat -A POSTROUTING -j snat --to-src aa:bb:cc:dd:ee:ff
exit 0
```

You can find background information on this in Dirk Hillbrecht's blog: *https://ipv6-first-guide.hillbrecht.de/#_ensure_correct_source_mac_address*.

32.5 Direct Access to the Contents of an Image File

This section is about how you can read or modify the contents of a virtual disk without starting the virtual machine itself. This is useful, for example, when you want to perform repairs from the outside or modify configuration files by means of a script.

There are different procedures for accessing the virtual disk:

- You can read or modify the contents of the virtual disks directly in the host system.
- You can make use of various `libguestfs` tools.
- You can use a Linux live system to help you—that is, start the virtual machine from the ISO image of a Linux distribution and access the virtual hard disks from there.

In this section, I will only discuss the first and second variants. Some of the tools presented here can only edit RAW images. You may need to convert an image file that is in a different format to this format. The `qemu-img` command helps you with this:

```
user$ qemu-img convert -f qcow2 image.qcow2 -O raw image.raw
```

The following command creates an equivalent RAW image from a logical volume or a hard disk partition:

```
root# dd if=/dev/mapper/vg1-lv2 of=copy.raw bs=64M
312+1 records in
312+1 records out
20971520000 Bytes (21 GB) copied, 133,462 s, 157 MB/s
```

Write access to a virtual disk is allowed only when the virtual machine is completely shut down! Otherwise you risk a corrupt file system!

32.5.1 Access to Partitioned RAW Images in the Host System

kpartx

You can use the `kpartx` command from the package of the same name to connect all partitions contained in a RAW file to loop devices:

```
root# kpartx -av image.raw
add map loop0p1 (252:12): 0 1024000 linear /dev/loop0 2048
add map loop0p2 (252:13): 0 19945472 linear /dev/loop0 1026048
```

In my tests, this also worked if the image file contained a GUID partition table (i.e., not a traditional partition table in the master boot record). If they are normal partitions, you can then `mount` them directly into the file system:

```
root# mkdir /repair1
root# mount /dev/mapper/loop0p1 /repair1
```

If you have configured LVM within the virtual machine, the resulting physical and logical volumes and volume groups are directly available. Lists of LVM elements are provided by `lvscan`, `pvscan`, and `vgscan`. The access to the LVs requires that the LVM tools are installed on the host system:

LVM

```
root# lvscan
ACTIVE            '/dev/VolGroup/lv_root' [7.56 GiB] inherit
ACTIVE            '/dev/VolGroup/lv_swap' [1.94 GiB] inherit
...
root# mkdir /repair2
root# mount /dev/VolGroup/lv_root /repair2
```

Now you can access the file systems of the virtual KVM hard disk via the `repairN` directories. When you're done with that, you need to clean up:

```
root# umount /repair1
root# umount /repair2
root# kpartx -dv image.raw
```

32.5.2 libguestfs Tools

Instead of analyzing the image file yourself and integrating the relevant partitions into the local file system, you can leave these tasks to various tools that are based on the `libguestfs` library.

The `libguestfs` library provides direct access to the file systems located within an image file. `libguestfs` supports all conceivable image formats and partitions, LVM, and all common file systems. The operation of the `libguestfs` tools is comprehensively documented on the following website and on the `man` page for `libguestfs`: *https://libguestfs.org*.

Current Debian, Fedora, RHEL, and Ubuntu distributions provide all the necessary packages right out of the box:

Installation

```
root# apt/dnf install libguestfs-tools libguestfs-tools-c
```

The package contains various commands for manipulating image files. You usually specify the image file to be processed or analyzed by using `-a imagefile`. Alternatively, you can specify the `libvirt` name of the virtual machine via `-d vmname`. `virt-df` provides a quick overview of the workload of all file systems of all virtual machines that are known to the `libvirt` tools. The command can also handle logical volumes inside virtual disks:

virt-df

```
root# virt-df
Filesystem                          1K-blocks       Used  Available  Use%
alma-mini:/dev/sda1                    495844      52091     418153   11%
alma-mini:/dev/sdb1                  20157836     253468   18880396    2%

alma-mini:/dev/vg_almamini/lv_root
                                      9571132    1051104    8033836   11%
alma64-vm2:/dev/sdb                   4319464    4319464          0  100%
alma64-vm2:/dev/sda1                   495844      32918     437326    7%
alma64-vm2:/dev/vg_vm2/lv_root        7539088    2369932    4786180   32%
...
```

If you are only interested in the results of a single virtual machine, you must specify the image file name with `-a` or the `libvirt` name of the virtual machine with `-d`.

virt-filesystems

`virt-filesystems` shows which file systems are contained in an image file. With the `--all`, `--long`, and `--uuid` options, the command also lists LVM and swap partitions and returns the UUID numbers of the file systems:

```
root# virt-filesystems -a vm2.img --all --long --uuid
Name                Type       VFS  Label Size        Parent  UUID
/dev/sda1           filesystem ext4 -     524288000   -       7a95...
/dev/vg_vm2/lv_root filesystem ext4 -     7843348480  -       e69f...
/dev/vg_vm2/lv_swap filesystem swap -     2113929216  -       e0f4...
...
```

virt-inspector

`virt-inspector` takes a look into the file systems of an image file and determines which distribution is installed in it, which partitions there are, which kernel version and which packages are installed, and so on. `virt-inspector` knows both the RPM and Debian package systems. The command returns an almost endless XML document as output, which is actually intended for further machine processing:

```
root# virt-inspector disk.img | less
<?xml version="1.0"?>
<operatingsystems>
  <operatingsystem>
    <root>/dev/cl/root</root>
    <name>linux</name>
    <arch>x86_64</arch>
    <distro>alma</distro>
    <product_name>Alma Linux release 7 (Core) </product_name>
    ...
    <mountpoints>
      <mountpoint dev="/dev/cl/root">/</mountpoint>
```

```
      <mountpoint dev="/dev/sda1">/boot</mountpoint>
    </mountpoints>
    <filesystems>
      <filesystem dev="/dev/cl/root">
        <type>xfs</type>
        <uuid>271158c8-8134-4082-9b74-f2413259ae4b</uuid>
      </filesystem>
   ...
```

virt-cat

`virt-cat` outputs a virtual machine file. You can use `-a` to specify the name of an image or `-d` to specify the (domain) name of the virtual machine:

```
root# virt-cat -d alma-server /etc/fstab
/dev/mapper/cl-root                     /     xfs  defaults 0 0
UUID=0112eac4-e995-4dfc-b6ca-4a5fc3b10d7b /boot xfs  defaults 0 0
/dev/mapper/cl-swap                     swap  swap defaults 0 0
...
```

To modify the file in an editor, you can use the `virt-edit` command instead of `virt-cat`. This command may only be used for virtual machines that are switched off! `virt-edit` takes the `EDITOR` environment variable into account when selecting the editor. By default, the `vi` editor is executed:

```
root# virt-edit -d alma-server /etc/fstab
```

virt-tar-in and virt-tar-out

You can use `virt-tar-out` to read a directory from the file system of an image file (`-a`) or a virtual machine (`-d`) and write it to a TAR archive:

```
root# virt-tar-out -d alma-server /etc etc.tar
```

If you also want to compress the archive, you must forward the output of `virt-tar-out` to `gzip` using a pipe:

```
root# virt-tar-out -d alma-server /etc - | gzip > etc.tgz
```

Conversely, `virt-tar-in` unpacks an archive in the image. The target directory must already exist. The virtual machine must be stopped beforehand:

```
root# virt-tar-in -d alma-server data.tar /home/user01
```

For compressed archives, you should proceed as follows:

```
root# zcat data.tgz | virt-tar-in -d alma-server - /home/user01
```

virt-make-fs

`virt-make-fs` creates a new image file and stores in it the contents of a directory or TAR archive, which may optionally be compressed:

```
root# virt-make-fs mydata.tar.gz new-disk.img
```

By default, the command creates a RAW image without a partition table with an ext2 file system. Options allow you to set a different image or file system format, MBR or GPT partitioning, control the size of the image file, and so on.

32.5.3 Converting an Image Format

qemu-img can convert disk images from one format to another. The following example creates a compressed QCOW2 image from a VDI file (VirtualBox). If you want to do without the time-consuming compression, you can simply omit the -c option:

```
user$ qemu-img convert -f vdi vm-disk001.vdi -O qcow2 -c vm.qcow2
```

After the conversion, you must make sure that the new image file has the correct owner or access rights so that the libvirt tools can access it.

32.5.4 Enlarging an Image

virt-resize

virt-resize copies a disk image and at the same time resizes the partitions and file systems it contains. The target image must be created beforehand. If the partition to be modified is used as a physical volume for LVM, virt-resize enlarges the PV via pvresize. You must run all other LVM commands yourself later in the running system. Prior to running virt-resize, you must shut down the underlying virtual machine!

In the following example, the first two commands analyze the source image. The third command sets up the enlarged target image. virt-resize then transfers the data from the source image to the target image:

```
root# qemu-img info alma.qcow2
file format: qcow2
virtual size: 10G (10737418240 bytes)
...
root# virt-filesystems --partitions --long -a alma.qcow2
Name       Type       MBR  Size         Parent
/dev/sda1  partition  83   524288000    /dev/sda
/dev/sda2  partition  8e   10212081664  /dev/sda
root# qemu-img create -f qcow2 alma-copy.qcow2 15G
root# virt-resize --expand /dev/sda2 alma.qcow2 alma-copy.qcow2
Summary of changes:
  /dev/sda1: This partition will be left alone.
  /dev/sda2: This partition will be resized from 9.5G to 14.5G.
             The LVM PV on /dev/sda2 will be expanded using
             the 'pvresize' method.
```

32.5.5 Reducing an Image File

While the previous section was about enlarging the image and subsequently the file system it contains, the opposite is often needed as image files often take up a lot of unnecessary storage space on the virtualization host.

Let's suppose you've set up a 20 GiB image file. `df -h` inside the virtual machine shows that the file system takes up only 10 GiB. Nevertheless, the image file on the host is much larger (possibly even larger than the intended 20 GiB). This is due to the fact that files deleted by the file system remain unused in the image file, virtually as garbage, and cannot be released at the host level.

virt-sparsify

Unnecessarily large image files not only slow down operation, but also make backups more laborious than necessary. The best way to resize an image is to use the `virt-sparsify` command. This command writes only the actually used blocks of the file systems contained in an image to a new image. The command requires that you stop the virtual machine first:

```
root# virt-sparsify image.qcow2 sparse-image.qcow2
```

Provided you trust `virt-sparsify`, you can also overwrite the image file directly:

```
root# virt-sparsify --in-place image.qcow2
```

Chapter 33
Windows Subsystem for Linux

Microsoft introduced a fundamentally new process in 2016: *Windows Subsystem for Linux* (WSL) enables you to install Linux distributions directly *in* Windows.

WLS and WSLg

In the first WSL version, Linux system functions were replaced by corresponding Windows functions. This allowed Linux programs to run without a real Linux kernel! This changed in 2019 with WSL2. Since then, WSL has been using a real kernel running through the Hyper-V virtualization technology. In most use cases, especially when accessing the file system, WSL2 produced a considerable improvement in performance.

The year 2021 saw the next major update, in which WSLg supported running Linux programs in graphics mode. Until now, WSL was only intended for text commands or server services. It was already possible to display the desktop or individual programs in graphics mode via RDP or VNC. Kali Linux, for example, makes use of this. However, WSLg promises a much higher performance because Linux programs get (almost) direct access to the graphics system via Wayland.

WSLg is an important piece of the puzzle for Windows 11. The new Windows version makes it possible to run selected Android apps directly. This feature has not been a resounding success so far, but the underlying technology is interesting in any case; behind the scenes, WSLg is used in conjunction with a new Android subsystem.

Target group

Aside from the Android apps, WSL is aimed at developers who want to run software from the Linux environment on Windows in an uncomplicated and efficient way. For Microsoft, WSL is an important piece of the puzzle in mitigating the exodus of developers to the open-source world. As far as I am concerned, WSL is certainly no reason to turn my back on "real" Linux—but if you primarily work on Windows, WSL can be an appealing alternative to virtual machines.

I personally find the technical implementation of WSL exciting, but in my experiments, I regularly had problems establishing reproducible network connections between the host (i.e., the Windows computer) and the WSL instances. Especially when WSL is to be used as a tool for developing server applications, the handling seems unnecessarily complex to me compared to Docker.

In a nutshell, for many use cases, it's easier and more practical to use Docker on Windows instead of WSL (or, of course, to switch to Linux altogether). Interestingly, Docker for Windows is itself based on WSL2. This clearly makes Docker one of the most important WSL applications.

Links You can find information on this topic at the following pages:

- *https://learn.microsoft.com/en-us/windows/wsl/*
- *https://github.com/microsoft/wslg*

33.1 Checking Out WSL

Installation WSL requires a 64-bit version of Windows 10 or 11. Windows Pro is not required. (On Windows Home, WSL2 uses a limited version of Hyper-V to run the Linux kernel.) To install WSL, you need to find the **Windows Features** program in the **Start** menu. In this program, you must enable the **Windows Subsystem for Linux** option (see Figure 33.1).

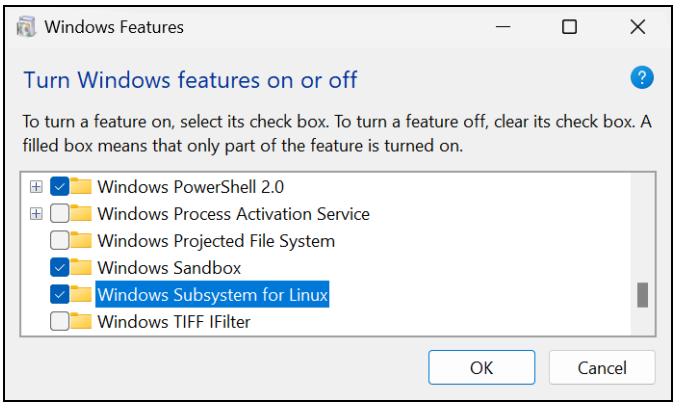


Figure 33.1 Installing WSL

Then you need to search for "Linux" in the Microsoft Store. There, AlmaLinux, Alpine Linux, Debian, Kali Linux, openSUSE, and Ubuntu were among the most recent choices. I did most of my testing with Ubuntu and Kali Linux.

Setting up a user The first time you start a WSL distribution, you must set up a user and password. This user has `sudo` privileges.

WSL installation for multiple Windows users If there are multiple accounts on your Windows computer whose users want to use WSL, the Linux installation must be repeated for each account. Each WSL distribution can only be used in the respective Windows account.

Uninstallation To remove a Linux distribution, open the **Apps • Installed Apps** dialog in the **System Preferences**, search for the distribution, and click **Uninstall**. Then, in

the terminal window, first run `wsl --list --verbose` and then `wsl --unregister name` for all distributions to be deleted.

33.1.1 Using WSL

By default, after starting a WSL distribution, you will be logged into the account that you specified during installation. Fortunately, the most important keyboard shortcuts work just like on Linux: Ctrl+D deletes the current character, Ctrl+A moves the cursor to the beginning of the line, and so on. You can now explore the new environment (see Figure 33.2). The `/etc/os-release` file shows the installed Ubuntu version.

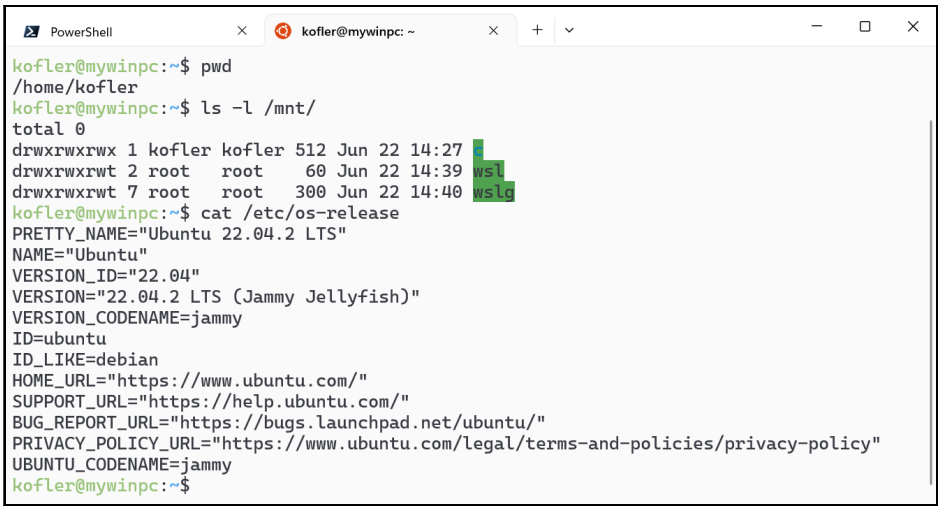


Figure 33.2 First Experiments with Ubuntu in WSL

Multiple WSL windows

You can open multiple WSL windows at the same time. These share a common process space and init process. As in "real" Linux, you can also see processes started in other windows via `ps ax` or `top` and terminate them using `kill`, if necessary. (Consequently, WSL windows are not comparable to separate Docker containers.)

> **Copying and Pasting Text**
>
> The WSL window uses the Windows terminal. The familiar Linux method of copying text with the mouse or trackpad and then pasting it with the middle mouse or trackpad button does not work in this way. Instead, you need to use a different approach. The mouse button enables you to select a text block. Enter then copies the text to the clipboard. Finally, you can use the right mouse button to insert this text at the current cursor position.

systemd

Since the end of 2022, WSL2 also supports systemd. In some distributions (such as Ubuntu), systemd is active by default. You can see this for yourself by using `systemctl`. For other distributions or past installations, you can enable systemd by adding the following lines to the local `/etc/wsl.conf` file within the distribution:

```
# in /etc/wsl.conf  (inside the WSL distribution)
[boot]
systemd=true
```

If it does not exist yet, you must create the `wsl.conf` file. The change does not become active until you exit the distribution, shut down WSL using the `wsl --shutdown` command in a Windows terminal, and then restart the distribution.

The major advantage of activating systemd is that you can now easily start web servers, SSH servers, or other services automatically within the WSL distribution:

```
user$ sudo systemctl enable --now apache2
```

Thus, as long as the WSL window of the distribution is open, the service in question is available on the local network.

Journal and Cron

When systemd is activated, the journal also runs (see Chapter 14, Section 14.13). This allows you to read system messages by using `journalctl`.

In most distributions, Cron is also enabled along with systemd. If that is not the case, you can simply install the appropriate package and activate the associated service via `systemctl`. Note, however, that Cron jobs are only executed when the relevant WSL window is open. Closing the WSL window ends the WSL execution with all background services, including Cron.

Package management

Within a WSL distribution, you can use common commands for package management; on Debian, Kali Linux, and Ubuntu, this particularly concerns the `dpkg` and `apt` commands. You can install new packages and perform updates.

User administration

The administration of users and groups works without any problems. You can set up new users via `adduser`, reset their passwords using `passwd`, change the `sudo` configuration by using `visudo`, and so on. When you start a WSL window, the default account is always applied, but you can change the active user via `su -l`.

Editors

In most distributions, the `nano` and `vi` editors are available for selection by default. Additional editors can be installed if needed.

Other commands

Common standard commands like `df`, `ifconfig`, `ip`, `less`, `locate`, `man`, `mount`, `ping`, `route`, `rsync`, `scp`, and `ssh` work without any problem.

However, if you want to run commands that directly call kernel or hardware functions, access devices, or rely on the contents of the `/proc` or `/sys` directories, the distribution must run with WSL2. This is true, for example, for the `iptables`, `nmap`, or `dmesg` command.

WSL2 is used by default in current installations. If necessary, you can check this in a Windows terminal by using `wsl --list --verbose`.

33.1.2 File System

WSL access to Windows files

To access the Windows file system from Linux, you need to use the `/mnt/c` directory. In the original WSL (hereafter referred to as WSL1), the `drvfs` driver is used, while WSL2 uses the `9p` network protocol.

In WSL2, all Windows file system files are assigned to the `root` user of WSL. Write permissions are only available for the own user directory (in this case, `C:\Users\ms`; on WSL, `/mnt/c/Users/ms`). Foreign user directories as well as system files and directories are read-only and partly even read-protected:

```
kofler$ cd /mnt/c/
kofler$ ls -l
ls: Config.Msi: Permission denied
ls: cannot access 'DumpStack.log.tmp': Permission denied
ls: cannot access 'hiberfil.sys':      Permission denied
ls: cannot access 'pagefile.sys':      Permission denied
ls: cannot access 'swapfile.sys':      Permission denied
ls: 'System Volume Information':       Permission denied
dr-xr-xr-x 1 root root    512 Jun 23 15:55 '$GetCurrent'
drwxrwxrwx 1 root root    512 May 29  2020 '$Recycle.Bin'
drwxrwxrwx 1 root root    512 Dec 11  2017 '$SysReset'
d--x--x--x 1 root root    512 Jun  8  2020  Config.Msi
...
kofler$ ls -l /mnt/c/Users/
ls -l /mnt/c/Users/
ls: /mnt/c/Users/backup-admin:         Permission denied
ls: /mnt/c/Users/git-test:             Permission denied
ls: /mnt/c/Users/martin-4321:          Permission denied
ls: /mnt/c/Users/test1:                Permission denied
ls: /mnt/c/Users/testuser:             Permission denied
lrwxrwxrwx 1 root root  18 May 22 15:02 'All Users' ->
                                        /mnt/c/ProgramData
dr-xr-xr-x 1 root root 512 Jun 24 11:35  Default
```

```
lrwxrwxrwx 1 root root  20 May 22 15:02 'Default User' ->
                                        /mnt/c/Users/Default
drwxrwxrwx 1 root root 512 Jun 24 12:19  Public
-r-xr-xr-x 1 root root 174 May 22 14:44  desktop.ini
d--x--x--x 1 root root 512 Jun 24 11:35  martin
drwxrwxrwx 1 root root 512 Jun 24 11:36  ms
d--x--x--x 1 root root 512 Jun 24 11:31  testuser
```

Windows access to WSL files

It's also possible to access WSL files from Windows the other way around. The easiest way to do this is by using Explorer. It contains entries in the sidebar for all installed WSL distributions (see Figure 33.3).

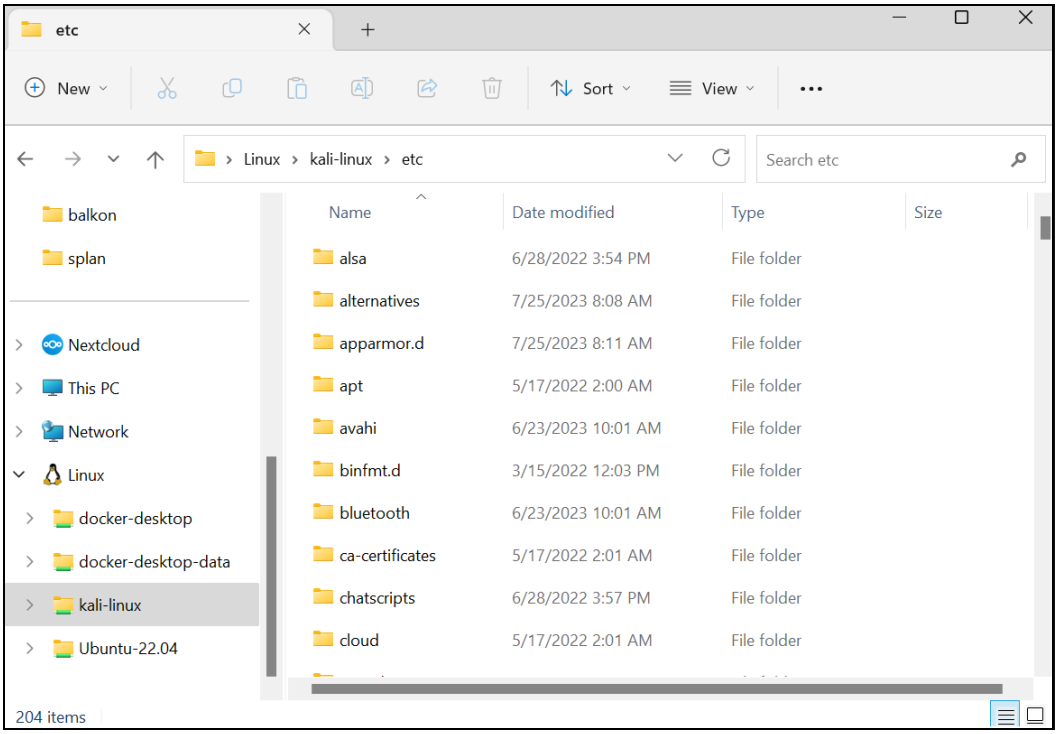


Figure 33.3 Access to Linux files in Windows Explorer

Internally, the WSL file system is mapped as a network directory. For example, the network name for the `/home/kofler` directory in the WSL distribution Ubuntu is as follows:

```
\\wsl.localhost\ubuntu\home\kofler
```

Own data

For WSL1, Microsoft recommended that frequently used files be stored outside of the actual Linux file system—in a subdirectory of `/mnt/c`—for efficiency reasons.

On WSL2, exactly the opposite rule applies. Because the Linux kernel is now natively responsible for file access, the files should be located within the distribution's file system, such as in `/home/<username>`.

The different recommendations are related to the fact that the storage location has changed. In the past, the Linux file system was mapped in a directory of the Windows file system. In WSL2, this is no longer the case. Rather, the Linux kernel running via Hyper-V has its own virtual `ext4` file system. This file system can grow up to a maximum of 256 GiB. You can find tips on how to expand the file system beyond this limit if necessary on the following page: *https://learn.microsoft.com/en-us/windows/wsl/compare-versions*.

33.1.3 Running Programs in Graphics Mode (WSLg)

Since 2021, you can also run Linux programs in graphics mode thanks to WSLg. In some distributions (such as Ubuntu), this works right from the start. Background information on the implementation of WSLg can be found on the GitHub project page at *https://github.com/microsoft/wslg*.

Usage

Once installed, the use of WSLg is very simple. You install the graphics program of your choice and launch it. The window appears on the Windows desktop:

```
user$ sudo apt install gedit
user$ gedit &
```

Alternatively, you can launch GUI programs from the Windows menu. You can find all installed programs under **Distribution Name • Program Name**—thus, for example, **Ubuntu • Text Editor** for `gedit`.

Practical experience

Basically, WSLg worked without any problems in my tests. Visually, the windows of Linux programs do not fit perfectly into the Windows world (see Figure 33.4), but program execution, operation, and the exchange of texts via the clipboard work without any problems. However, programs that are not yet optimized for Wayland, such as GIMP, react quite sluggishly to attempts to change the window size.

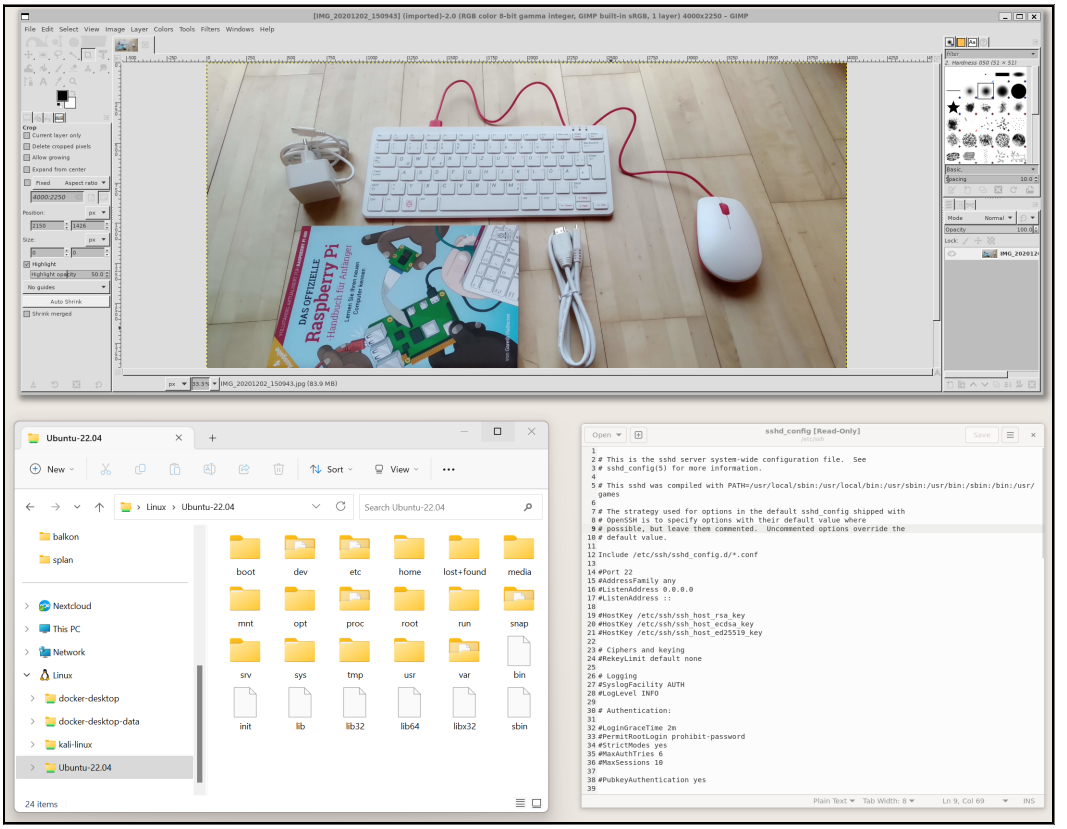


Figure 33.4 GIMP as Linux Program (Top); Windows Explorer and "gedit" Peacefully United on Windows Desktop (Bottom)

No Linux desktop...

WSLg can run individual Linux programs directly on the Windows desktop. However, the current implementation is not intended to run an entire Linux desktop like GNOME. Perhaps this feature will follow in the next WSL version?

... except in Kali Linux

It's already possible to use the Linux desktop via VNC on Windows. In most distributions, however, this requires relatively complicated configuration work. But Kali Linux makes it much easier for you: After installing the `kali-win-kex` package you need to run the `kex` command (without `sudo`!). This command sets up a VNC server and first asks you twice for a password for the VNC connection. Optionally, you can set up another password for a read-only operation, but this is rarely useful:

```
user$ kex
Password: *******
Verify:   *******
Would you like to enter a view-only password: n
```

After that, Kali Linux appears in full screen mode. F8 takes you to a context menu of the VNC viewer. There you can not only disable the **Full Screen** option, but also minimize the VNC window or end the session altogether (**Exit Viewer**). The Kali desktop automatically adjusts to the window size of the VNC viewer.

33.2 WSL Network Integration

WSL1: native integration

The WSL network integration depends on the WSL version you use. WSL1 shares the Windows network with Linux. This means that the WSL distribution has the same IP address and host name as Windows.

WSL2: NAT

WSL2, on the other hand, receives its own IP address in a private address space. The behavior is strange when several distributions are running at the same time as these distributions then share the IP address. The IP address is on a private network and is randomized each time WSL is started. The connection to the host system is made via NAT—basically the same as for virtual machines. A network connection between the Windows computer and the WSL system is possible (in both directions). On the other hand, integration of WSL into the local network is impossible:

```
user$ ip addr show eth0
3: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq ...
    link/ether 00:15:5d:4c:fd:62 brd ff:ff:ff:ff:ff:ff
    inet 192.168.3.193/16 brd 192.168.15.255 scope global eth0
       valid_lft forever preferred_lft forever
    ...
user$ ip route
default via 192.168.0.1 dev eth0
192.168.0.0/20 dev eth0 proto kernel scope link src 192.168.3.193
```

A major problem in the practical use of WSL is that Windows does not know the WSL host names. Direct network connections therefore require the specification of IP numbers—and these change at the latest with the next restart of Windows. Thus, if you are running an HTTP server on Linux, you must first use `ip addr` to determine the IP address of Linux before you can specify this address as *http://192.168.3.193* in Edge or another web browser.

After all, Linux can address the local Windows host as `<hostname>` or `<hostname>.local`.

WSL2: host forwarding

WSL2 provides the option to forward the WSL network to the network of the Windows host. Thus, WSL2 behaves quite similarly to WSL1. Network services run by Linux are visible and accessible under `localhost` on Windows.

To enable this host-forwarding feature, you need to create the `.wslconfig` configuration file in your Windows home directory (usually `C:\Users\<name>`) with the following content:

```
# file .wslconfig in the Windows home directory
[wsl2]
localhostForwarding=true
```

To activate the setting, you must restart WSL. To do this, you can run the following commands in the terminal, PowerShell, or `cmd.exe`:

```
> wsl --shutdown
> wsl                    (starts the default WSL distribution)
```

The web pages of a web server running Linux can then be easily viewed in a web browser running Windows at *http://localhost*.

Note that this approach has serious disadvantages:

- **Port conflicts**
  Windows and WSL now share all ports. For example, it is not permissible for a web server to claim port 80 on both Windows and Linux/WSL.
- **Incompatible with Docker**
  Due to the `localhostForwarding=true` setting, Docker for Windows no longer works.
- **Problems with Windows restart**
  Some WSL users complain that the `localhostForwarding` option works unreliably when Windows is restarted. The remedy after the restart is an explicit WSL restart (i.e., `wsl --shutdown`) or disabling Windows Quick Start (see *https://github.com/microsoft/WSL/issues/5298*).

The integration of WSL2 into the local network therefore leaves a lot to be desired. Accordingly, some WSL users have formulated various suggestions for improvement on GitHub. On the following pages, you will also find ideas for how to get around the WSL restrictions by automatically running scripts and other hacks for emergencies:

- *https://github.com/microsoft/WSL/issues/4305*
- *https://github.com/microsoft/WSL/issues/5780*
- *https://github.com/microsoft/WSL/issues/5812*

33.3 The wsl Command and WSL Configuration

wsl command

The `wsl` command is available in the Windows terminal. Without any additional parameters, it starts the WSL distribution directly in the currently

active window. Then you can use WSL until you terminate the session using `exit` or Ctrl+D. If multiple WSL distributions are installed, the default distribution will be used. This can be set via `wsl --set-default`.

The `-d <wslname>` and `-u <username>` options allow you to specify for which WSL installation and for which user the command should be run. From a security perspective, the possibility of gaining `root` privileges without any protection is confusing. (In particular, it is not necessary to specify a password or use `sudo`):

```
> wsl -d kali-linux -u root
root# cat /etc/shadow
...
root# exit
```

Running Linux commands

Instead of using `wsl` interactively like a shell, you can also use `wsl` to run a single command, again using the `-d` and `-u` options:

```
> wsl cat /etc/os-release      (run command in default account)
> wsl -u root passwd kofler    (run command with root privileges)
```

WSL administration

`wsl` also supports you with admin tasks. You can use it to list WSL distributions and switch them from WSL1 to WSL2 (and back again if necessary):

```
> wsl --list  --verbose
  NAME            STATE           VERSION
  Ubuntu          running         2
  kali-linux      running         1
> wsl --set-version kali-linux 2
```

Finally, here's a brief overview of other commands:

- `wsl -t <wslname>` shuts down the specified WSL distribution.
- `wsl --shutdown` terminates all running distributions. The next time WSL is started, a new IP address will be used.
- `wsl --unregister <wslname>` deletes the specified distribution without prompting. The image downloaded from the Microsoft Store remains intact, but the distribution will be completely set up again the next time it is started. All your own files and settings will be lost. `wsl --unregister` is therefore a kind of total reset for the distribution in question.
- `wsl --help` displays even more commands.

33.3.1 Global WSL Configuration

.wslconfig

In the home directory of your Windows computer, usually in *C:\Users\<name>*, you can set up the `.wslconfig` configuration file using an editor

and define some global WSL options there. By default, the file does not exist. The following listing provides some examples. Don't forget to prefix the file with `[wsl2]`:

```
# file .wslconfig in the Windows home directory
[wsl2]
# WSL should use only 4 cores (default: all cores)
processors=4

# WSL should use max. 6 GiB of RAM (default: 50% of RAM)
memory=6GB
```

The settings will not take effect until WSL is restarted (i.e., you must run `wsl --shutdown`). A reference about the currently available options can be found here: *https://docs.microsoft.com/en-us/windows/wsl/wsl-config*.

33.3.2 Linux-Specific Configuration

/etc/wsl.conf

In addition to the global `.wslconfig` configuration file just mentioned, the `/etc/wsl.conf` file specific to this distribution can also be set up in each WSL installation. Note that this file must be included in the Linux file system and not in the Windows file system! The file is analyzed when the distribution is started. The following lines show examples of some frequently used options:

```
# file /etc/wsl.conf in the file system of the Linux distribution
# Mount C:, D: etc. in Linux as /mnt/c, /mnt/d etc.
# (enabled = true applies by default)
[automount]
enabled = true
root = /mnt/

# automatically create /etc/hosts and /etc/resolv.conf
# (generateHosts = true applies by default)
[network]
generateHosts = true
generateResolvConf = true

# enable systemd
[boot]
systemd=true
```

More information about the setting options in `wsl.conf` can be found on the following pages:

- *https://docs.microsoft.com/en-us/windows/wsl/wsl-config*
- *https://devblogs.microsoft.com/commandline/automatically-configuring-wsl*

33.3.3 Enlarging the WSL2 Disk

For each WSL installation, WSL2 creates a disk image with an `ext4` file system that may be up to 256 GiB in size. The associated `*.vhdx` file is located in the following directory:

C:\Users<username>\AppData\Local\Packages<wslname>\LocalState

This file can be enlarged using the `diskpart` Windows partitioning tool if necessary, provided WSL is stopped first. The most tedious point here is entering the full path to the `*.vhdx` file. The best way is to find the file in Windows Explorer and then copy the path from there to your terminal. You need to specify the required size in MiB:

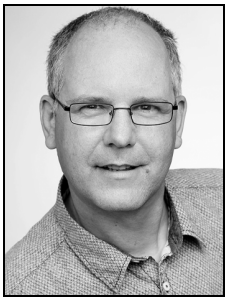
```
> wsl --shutdown
> diskpart
DISKPART> select vdisk file="C:\pfad\zu\LocalState\ext4.vhdx"
DISKPART> detail vdisk
Virtual size: 256 GB
...

DISKPART> expand vdisk maximum=500000   (also 500 GiB)
DISKPART> exit
```

Saving storage space

If you use WSL intensively, the virtual machines created in the process can take up a lot of space on your disk. The easiest thing to do then is to simply delete the respective WSL distributions and reinstall them. If you don't want that, check the following page for tips on cleaning up without losing data: *https://nickjanetakis.com/blog/reclaiming-tons-of-diskspace-by-compacting-your-docker-desktop-wsl-2-vm*.

The Author



Dr. Michael Kofler is a programmer and Linux administrator. He studied electrical engineering/telematics at Graz University of Technology. He has been one of the most successful and versatile computing authors in the German-speaking world for many years. His current topics include Linux, Docker, Git, hacking and security, Raspberry Pi, and the programming languages Swift, Java, Python, and Kotlin. Michael Kofler also teaches at the Joanneum University of Applied Sciences in Kapfenberg, Austria.

Index

? (wildcard character) 209, 260
.cache directory 159
.config directory 159
.desktop files 588
.htaccess file (Apache) 858
.local directory 159
(( )) (arithmetic expressions) 211
(strings) 213
\\ (strings) 213
[INCLUDES] 854
*.exe file 297
* (wildcard character) 209, 260
** (wildcard character) 210, 249
/bin 290
/boot
  /efi 51, 688
  /grub 696
  /initrd 689
  /initrd, creating yourself 692
  /vmlinuz 689, 755
/dev 290
  /disk 602
  /mapper 668
  /md 656
  /sd 599
  /vd 600
  internals 293
  list 294
/etc 290, 394
  /adduser.conf 403
  /aliases 899, 909, 924
  /alternatives 545
  /apparmor.d 1078
  /apt/apt.conf 528
  /apt/sources.list 528
  /bashrc 196
  /boot/grub.cfg 696
  /chrony.conf 400
  /cockpit 465
  /cron.daily 320
  /cron.hourly 320
  /cron.monthly 320
  /cron.weekly 320
  /crontab 318
  /crypttab 681
  /cups 447
  /default/grub 698
  /default/language 419
/etc (Cont.)
  /deluser.conf 403
  /dhcp/dhclient.conf 498
  /dnf/dnf.conf 514
  /dovecot 932
  /dracut.conf 693
  /firewall.d 1051
  /fstab 619, 624
  /fstab (CIFS) 1003
  /fstab (SSD TRIM) 676
  /group 405
  /gshadow 411
  /host.conf 488
  /hostname 490
  /hosts 487
  /hosts.allow 1046
  /hosts.deny 1046
  /inittab 727
  /inputrc 196
  /libvirt 1110
  /lightdm 588
  /locale.gen 423
  /localtime 398
  /login.defs 284, 409
  /logrotate.conf 457
  /mdadm/mdadm.conf 657
  /modprobe.conf 479
  /modules 741
  /mtab 617
  /Muttrc 347
  /my.cnf 876
  /network/interfaces 495
  /NetworkManager/system-connections 493
  /nscd.conf 418
  /nsswitch.conf 416
  */PackageKit/** 541
  /pam.conf 413
  */pam.d/** 413
  /passwd 404
  /php.ini 868
  /polkit-1 313
  /postfix 909
  /printcap (CUPS) 447
  /profile 196, 215, 216
  */rc.d/** 727
  /rc.d/rc.local 730
  /rc.local 731

/etc (Cont.)
/resolv.conf ... 489
/resolv.conf (Ubuntu) ... 490
/rsyslog.conf ... 452
/samba/smb.conf ... 969
/selinux ... 1075
/shadow ... 407
/shells ... 195
/skel ... 405
/smartd.conf ... 674
/ssh ... 810
/ssl/certs ... 472
/sudoers ... 307
/sysconfig/console ... 397
/sysconfig/i18n ... 419
/sysconfig/language ... 419
/sysconfig/network ... 490
/sysconfig/network-scripts ... 492
/sysctl.conf ... 765
/systemd/journald.conf ... 462
/systemd/network ... 501
/timezone ... 398
/updatedb.conf ... 268
/vconsole.conf ... 396
/wpa_supplicant ... 485
/wsl.conf ... 1140, 1148
/xdg/user-dirs.conf ... 159
/yum.conf ... 514
/zsh ... 243
/home ... 290
/lib
/modules ... 736, 737, 754
/media ... 291
/mnt ... 291
/opt ... 291
/proc ... 291, 759
/asound ... 432
/config.gz ... 751
/crypto ... 679
/mounts ... 617
/pci ... 427
/sys ... 765
/root ... 291
/run ... 291
/log/journal ... 460
/sbin ... 291
/init ... 726, 727
/share ... 291
/srv ... 292
/www ... 830
/sys ... 292, 760
/kernel/security ... 1078
/tmp ... 292
/usr ... 292
/var ... 292
/lib/dpkg/alternatives ... 547
/lib/rpm/alternatives ... 547
/log/journal ... 460
/log/Xorg.0.log ... 584
/run ... 301
/spool/cron/tabs ... 318
/www ... 830
& (background processes) ... 298
#! (command interpreter) ... 217
< (output redirection) ... 203
> (input redirection) ... 203
~ (home directory) ... 254
~/bin ... 224
$() (command substitution) ... 211
$ (variables in bash) ... 213
2FA (two-factor authentication) ... 818
389 Directory Server ... 392
4K displays
GNOME ... 144
KDE ... 172
4K screens, VirtualBox ... 1101
7zr ... 1020

A

a2disconf ... 833
a2dismod ... 833
a2enconf ... 833
a2enmod ... 833
a2ensite ... 833
AAAA record (DNS) ... 902
aa-complain ... 1079, 1080
aa-enforce ... 1079, 1080
aa-status ... 1078
Abbreviations ... 201
Access bits ... 274
for new files ... 283
setuid, setgid ... 279
sticky ... 281
Access control ... 401
Access control list (ACL) ... 285
Access point ... 473
Access rights, basic principles ... 273
acme.sh ... 845
ACPI ... 425
kernel boot options ... 764
Action (syslog) ... 454
Active Directories ... 967
Adaptable Linux Platform (ALP) ... 99

addgroup 403
addr
addr 480
link 479
adduser 403
Administration 391
Administrator account 69
akmods 1089
akms 745
alias 201
alias_database 924
alias_maps 925
Alias (email) 924
Alias (httpd.conf) 853
alias (in modprobe.conf) 741
Allow (Apache) 855
Allowed-Origins 534
Allow-hotplug, in /etc/network/
interfaces 495
AlmaLinux 41, 783
alsactl 432
alsamixer 432, 433
alternatives 546
Amazon Linux 798, 807
Amazon Web Services (AWS) ... 795, 1032
amdgpu 573
amdgpu-pro 574
anacron 322
Android 37, 41
AnyDesk 148
Apache 829
authentication 857
FPM/FastCGI 869
HTTPS 836
MPM 869
password 857
securing a directory 857
SELinux 831
Unicode 834
virtual hosts 859
apfs file system (Apple) 613
aplay 432
AppArmor 1069, 1077
apparmor-utils 1079
AppImages 552
AppIndicator (GNOME extension) 154
Apple
CUPS 444
zsh 241
Apple file system 613
Applets
GNOME 122
Applets (Cont.)
KDE 163
applydeltarpm 511
AppStream 518, 780
APT 527
automatic updates 533
apt 530
apt-cache 533
apt-daily-upgrade 533
apt-get 528
aptitude 528
apt-key 529
apturl 533
arch-chroot 705
Archiving files 1019
GNOME 138
Arch Linux 41, 95
package management 537
arecord 432
A record (DNS) 902
Arithmetic expressions (bash) 211
arptables 1059
ASCII 418
Asymmetric encryption 837
ATAPI see IDE 598
audit.log (SELinux) 1076
AUR packages 539
AuthConfig 854
authconfig 416
Authentication
Apache 857
POP/IMAP 937
authselect 416
Authy 821
autocd 245
autojump 256
Auto login 588
KDE 172
SUSE 589
automatic.conf 521
Automatic execution of jobs 318, 323
Autostart
GNOME 157
KDE 172
Avahi 502
avahi-browse 503
avahi-daemon 502
avahi-discover 503
avahi-dnsconfd 503
Awesome Tiles (GNOME
extension) 154
AWStats 861

B

Background processes 298
backintime 1010
Backup file (Emacs) 371
Backups
  incremental 1025
  KVM 1032
  LVM snapshots 1031
  MySQL 888
Backup script 220, 221
baobab 138
bash 193
  completion 199
  configuration 195
  keyboard shortcuts 200
  programming 217
  shell types 196
  variables 226
bash_history 198
bashrc 91, 196
bash script examples 325
bat 272
Battery (notebooks) 425
bg 299
Binary package 510
bind interfaces only 973
BIOS 50
  system boot 691
  updates 541
BIOS GRUB partition 692
Blacklist (email) 902
blacklist (modprobe.conf) 742
blkid 630
blkid -s PART_ENTRY_TYPE 624
Bluetooth 427
bluetoothd 428
boltctl 431
Bonjour 502
Bookworm 77
bootctl 710
Bootloader 687
  systemd-boot 707
Boot options 760
Boot partition 65
Boot process
  bootloader 691
  System-V init 726
Borg Backup 1015
Boxes 1105
Brace expansion 210
Branches (bash) 230
Bridged networking 1097
bridge-utils 1126
browseable 983
Browsing (Samba) 964
BSD license 45
btrfs file system 631, 632
  Fedora 645
  RAID 638
  Snapper/openSUSE 642
  swap files 653
Buckets (S3) 1033
Budgie 110
Bullseye 77
bunzip2 1019
Buster 77
bzip2 ..... 1019, 1020

C

Calamares installer 79
  Linux Mint 93
Calendar (GNOME) 141
Cannot change locale (error message) 423
Canonical 43
canonical-livepatch 757
Capabilities 289
case 233
cat 185
ccat 252
cd, zsh 245
CentOS 41, 781
  Stream 782
certbot 845
Certificate
  creating and signing by yourself 838
  HTTPS (Apache) 837
  POP/SMTP (Dovecot) 938
  snakeoil certificate 843
chage 408
Changing global settings 765
Character set 418
  Apache 834
  PHP 834
chattr 633
checkarray 658, 662
chpasswd 410
chrome-gnome-shell 153
Chrome OS 41
Chrony 400
chroot 705, 1048
chsh 194, 242

cifs file system 1000
cifs-utils 1001
ClamAV 947
clamav-milter 948
classes.conf 447
Clear Linux, /etc/fstab 625
cless 252
Client configuration 469
Clipboard 128
KVM/libvirt 1118
VirtualBox 1099
cloneconfig 751
Cloud, server installation 773
cloud-guest-utils 804
cloud-utils-growpart 804
Cluster SSH 824
cmdline file 759
Cockpit 392, 463
managing virtual machines 1114
cockpit.conf 465
cockpit-machines 1114
CodeReady Linux Builder 560
colored-man-pages 252
colorize 252
Color profiles 145
Command 211
Commands 297
command interpreter 193
executing 205
executing conditionally 207
executing in the background 206
executing regularly 318, 323
input 198
starting 298
starting (bash) 206
substitution (bash) 211
Comparisons (bash) 231
Comparisons of numbers (bash) 231
complete 199
compsize 645
compsys 244
Computer startup 687
Conditions (bash) 231
Configuration 391
file system 619
font 397
kernel 745, 750
keyboard (text console) 395
LAN 469
network 469
password 409
prompt 196
Configuration (Cont.)
setting up users 402
text console 395
time zone 397
conky 301
Console
changing 179
font 397
keyboard 395
console-setup 395, 397
Contacts (GNOME) 141
Contrib packages 554
Copr (DNF) 520
Copy on write (COW) 630
deactivating 633
CoreOS (Fedora) 85
Cosmic DE 104, 105
cp 255
changing the name during the copy process 262
cPanel 392
CPU
determining the current clock frequency 435
governor 436
temperature 437
tuning 434
turbo boost 436
cpu-checker 1106
cpupower 435
cracklib 409
CRB repository 560
create mask 985
Cron 318
replacing with systemd 323
WSL 1140
crontab 318
cryptsetup 677
csh 194
CSSH 824
ctrl-alt-del.target 716
CUPS 444
internal details 447
cupsd 447
cupsd.conf 447
curl 344, 1032
custom.conf 589

D

daemon.conf 589
Daemons 315

Dash 124
dash 195, 218
Database server 873
Data partition 65
dbus-daemon 430
dconf 156
dconf-editor 156
dctrl format 525
dctrl-tools 525
dead keys 395
Debian 42
firmware 78
initrd file 693
keyboard 395
package management 524
server installation 793
static network configuration 495
system startup 730
VirtualBox 1096
declare 215
Déjà Dup 1007
delgroup 403
deltarpm 511
Delta updates 511
deluser 403
Deny (Apache) 855
DenyHosts 812
Desktop
GNOME 119, 126
KDE 161
DeviceKit 430
Devices 276, 293, 598
internal details 293
kernel modules 741
udev file system 294
Devuan 720
df 616
dhclient 482, 498
dhclient.conf 498
dhcpcd 482
Directory 254
basic principles 254
tree 289
directory mask 985
Directory Server 392
Directory tree, partitions 594
disable_vrfy_command (Postfix) 931
Disabling the CapsLock key 139
Disabling the touchpad 140
discard 621, 676
LUKS encryption 681
Discover 541
firmware updates 543
Discoverable Partitions
Specification 624
Disc quotas 592
Disk images (libvirt/KVM) 1112
Disks (gnome-disks) 610
Dispatcher (NetworkManager) 478
Display manager 587
Distributions
Linux 39
overview 41
updates 70
dis-ucode-ldr (boot parameter) 545
DKIM 951
dkms 743
DKMS, VirtualBox 1089
DLNA, GNOME sharing 136
dm_crypt 677
DMARC 959
dmesg 455
DNF
automatic updates 521
package manager 513
dnf 513
dnf.conf 514
dnf-automatic 521
dnf-makecache
internal details of systemd timer 324
dnf-plugin-system-upgrade 556
dnf-utils 521
DNS
client configuration 489
mail server 902
Reverse DNS 905
DNS resolver (Ubuntu) 490
DocumentRoot (Apache) 830
Documents directory 158
Dolphin 167
sharing a directory 989
DomainKeys Identified Mail 951
Domain-level security 966
do-release-update 565
dotglob 210
Dovecot
IPv6 936
POP/IMAP authentication 937
SMTP authentication 938
dovecot.conf 932
Downloads directory 158
dpkg 526
examples 526

dpkg (Cont.)
status code 525
dpkg-reconfigure 399, 527
dracut 693
DRI 571
Drive letters (C:, D:) 594
Driver installation (Ubuntu) 564
DRM 569
Dual licenses 45
duf 272
duplicity 1027

E

e2label 629
ebtables 1059, 1129
EDITOR 188
Editors 186
Emacs 369
joe 188
nano 186
Vim 351
edk2-ovmf 1118
EFI 50
GPT 687
GRUB 688
in KVM/QEMU 1118
partition 51, 688
secure boot 51, 690
server installation 777
systemd-boot 707
system partition 707
system partition (ESP) 51
system startup 687
updates 541
efibootmgr 706
example 778
EGL 569, 571
Elastic Block Store (EBS) 799
Elastic Cloud service (EC2) 795
Elastic IP (AWS EC2) 805
Electronic Frontier Foundation 844
Elementary OS 110
ELinks 346
Elvis 187
Emacs 369
.emacs file 386, 387
automatic backup 371
buffer 382
colored text 385
configuration 386
cursor movement 373
Emacs (Cont.)
dynamic shortcuts 379
editing commands 375
editing modes 372, 384
extensions 388
font-lock-mode 385
indenting and outdenting 376
MELPA 388
online help 371
quick start 186
regular expressions 380
search and replace 381
searching 379
setting the font 386
syntax highlighting 385
tabs 376
windows 383
Email 895
alias 924
basic principles 895
blacklist 902
DNS 902
mutt 346
relaying 902
server 895
viruses 947
Emergency
repairing the file system 626
Emergency target 716
Encryption 62
files 676
file systems 677
mail server 901
EndeavourOS 41
Energy saving functions 425
ENI configuration 495
Enlarging the Linux file system 1103
Environment variables 215
EPEL package source 560
erd 273
ESP (EFI System Partition) 707
Etcher 53
etckeeper 394
Ethernet controller
configuring 478
ethtool 491
evim 367
exa/eza 271
ExecCGI 854
Exec Shield 1070
ExecStart keyword (systemd) 722
exFAT file system 648

exfat-utils 648
exit (bash) 224
Expansion mechanisms (bash) 208
expect 410
Experimental packages 555
export 215
ext4 file system
Windows access 630
Extended attributes 285
extendedglob 245
Extended partition 59
Extension pack (VirtualBox) 1090
ext file system 627

F

f.lux program 145
FAI (Fully Automatic Installation) 392
faillock 411
faillog 411
Fallout 765
Fan control 443
FAT file system 647
fc-list 149
fd 273
Fedora 42, 84
distributions update 556
dracut 693
firewall 1051
initrd file 693
keyboard 396
static network configuration 492
sudo 311
system startup 729
fetchmsttfonts 149
fg 299
fglrx 573
FIFO 204
FileInfo 854
File management, basic principles 253
File manager Nautilus 128
File name expansion 198
Files
basic principles 253
compressing 1019
copying using sed 262
finding 266
names 253
printing 451
wildcard characters 209
File system
check 626
File system (Cont.)
configuration 619
enlarging (ext3) 630
ext2 file system 620
ext3 file system 627
managing 591
maximum file size 627
repair 626
types 611
virtual 614
WSL 1141
Filesystem Hierarchy
Standard (FHS) 289
File type in ls command 257
Filters
IP packet filters 1049
printing 446
find 268
Finding
files 266
findandgrep 268
findmnt 617
Firewall 1039
AWS EC2 806
example 1066
mail server 908
openSUSE 103
packet filters 1049
Samba 972
web server 829
firewall-cmd 908, 1053
FirewallD 1051
Amazon Linux 807
firewalld 1051
Firmware (Debian) 78
Firmware updates 541
fish (Konqueror) 170
Flatpak 549
FollowSymLinks 854
Font 148, 419
Emacs 386
text consoles 397
font-lock-mode 385
for (bash) 233
forcefsck 626
force group = +sales 984
Foreground processes 298
Foreshadow 765
Forgotten root password 410
Fork type (systemd) 722
FORMAT (Windows) 58

Formatting
btrfs file system ... 632
exfat file system ... 595, 648
ext3/ext4 file system ... 628
ntfs file system ... 595, 648
vfat file system ... 595, 648
xfs file system ... 646
Formatting data media ... 595
FPM/FastCGI method ... 869
Fractional scaling ... 144
free ... 425
Free Software Foundation ... 44
freshclam ... 947
fsck ... 626
fsck.ext2/ext3/ext4 ... 629
fsck.xfs ... 647
FSF ... 44
FSSTND ... 289
fstab ... 619
CIFS ... 1003
fstrim ... 676
LUKS encryption ... 681
FTP ... 341
client ... 341
secure FTP server ... 810
server (sftp ... 810
ftp command ... 341
function (bash) ... 236
fuser ... 301
Fuzzy finder ... 272
fwupd ... 542
fwupdmgr ... 542
fx ... 273
fzf ... 272, 273

G

Gateway
client configuration ... 489
client configuration (route) ... 481
gdisk ... 624, 664
gdm ... 587, 588
General Public License (GPL) ... 44
getcap ... 289
getfacl ... 286
getfattr ... 288
getsebool ... 1073
getsll ... 845
GID ... 405
GL (Open GL) ... 571
glibc, time zone ... 398
Global Filesystem (GFS) ... 593
Globbing
bash ... 208
zsh ... 247
globstar ... 210
glow ... 273
GLX ... 571
glxinfo ... 583
glx-utils ... 583
GMT (Greenwich Mean Time) ... 397
GNOME ... 119
extensions ... 152
gdm ... 588
proxy settings ... 474
sharing a directory ... 988
shell extensions ... 152
shell themes ... 156
Thunderbolt ... 431
Tweak Tool ... 150
Wayland ... 569
gnome-browser-connector ... 153
gnome-disks ... 610
gnome-disk-utils ... 610
gnome-extensions ... 152
gnome-extensions-app ... 152
gnome-font-viewer ... 149
gnome-keyring-daemon ... 816
gnome-language-selector ... 419, 565
gnome-nettool ... 333
gnome-session-properties ... 157
gnome-software ... 541
firmware updates ... 543
gnome-startup-applications ... 157
gnome-system-monitor ... 301
gnome-terminal ... 181
gnome-tweak-tool ... 139, 150
gnome-user-share ... 136
gnome-vfs.keys ... 158
gnome-vfs.mime ... 158
GNU ... 46
Emacs ... 369
General Public License ... 44
GRUB ... 687
GoAccess ... 862
Google Analytics ... 862
Google Authenticator ... 818, 819
google-authenticator ... 819
Governor (Intel CPUs) ... 436
GParted ... 610
gpasswd ... 411
gpg ... 676, 1036
GPL, Apple ... 241

GPT 604
BIOS GRUB installation 692
EFI 687
partition numbers 601
graphical target 586
grep 270
grepall 217
grep-dctrl 525
groupadd 403
Groups 405
new files 283
of files 273
growpart 804
Grsync 1013
GRUB 687
configuration 696
hard disk names 702
operation 695
partition names 702
Secure Boot 690
server installation 776
Version 2 696
grub.cfg 696
grub2-mkconfig 696, 698
grubby 687, 700
gsettings 156
GStreamer 433
GTUBE test message 943
gucharmap 150
guest account 985
guest ok 986
guest only 986
GUID partition tables (GPT) 59
gummiboot project 708
gunzip 1019, 1020
Gutenprint 446
GVFS 134
gvim 351
gzip 1019, 1020

H

Hacker kernel 747
halt 725
Hard disk
formatting 58
partitioning, Linux 64
Hard links 264
Hardware
devices 293
devices (udev) 294
reference 425
Hardware Enablement Stack 114
Hardware RAID 654
hashbang 218
hd0,0 (GRUB) 702
help 190
Heredoc syntax 237
SSH 337
HFS file system (Apple) 613
Hibernation mode 426
forcing a restart 726
hidden_host 347
HiDPI displays
GNOME 144
KDE 172
HiDPI screens
VirtualBox 1101
History
APT 533
DNF 516
ZYpp 524
History (bash) 198
hold status (dpkg command) 527
Home directory 254, 404, 405
Home server 990
HOME variable, sudo on Ubuntu 310
host.conf 488
Host name
server installation 773
setting 490
SUSE 103
hostnamectl 720
Host RAID 654
hosts 487
hosts.allow 1046
hosts.deny 1046
hosts allow 973
hosts deny 973
Hotplug system 430
HPLIP 449
hplip-gui 449
hplip-toolbox 449
HP printer driver 449
htop 300
htpasswd 857
HTTP
proxy 474
web server 829
httpd 829
httpd.conf 832
HTTPS 836
hwclock 398
HWE packages 114

Hybrid graphics (NVIDIA) 580
hydra 812
Hyperthreading 767
Hyper-V 1091

I

i18n 418
i915 574
ICMP 1041
Icons (GNOME) 121, 155
Icon themes 156
IDE hard disks 598
IdentityFile 817
idn 491
if (bash) 231
ifcfg-rh 492
ifcfg-rh (NetworkManager plugin) 477, 492
ifconfig 480
ifdown 498
IFS 220
ifupdown (NetworkManager plugin) 477
Image directory 158
Images
converting the format 1134
enlarging (QCOW2) 1134
formats 1112
libvirt/KVM 1110
read/manipulate 1130
VirtualBox 1093
IMAP server authentication 937
Immunix 1077
Incremental backups 1025
Indexes 854
inet_interfaces (Postfix) 911
inet6 496
iNet Wireless Daemon 487
info 190
init 726
Initial RAM Disk 692
Init process log 456
initrd (GRUB) 702
initrd file 689
creating yourself 692
Init system 711
Debian 730
Fedora 729
Raspberry Pi OS 731
RHEL 729
Ubuntu 730
inittab 727
Init-V process 726
kernel parameters 765
Init-V scripts 727
InnoTek 1087
inputrc 196
Input redirection 202
sudo 309
install (in modprobe.conf) 742
Installation 56
basic configuration 68
basic principles 49
external media 55
instructions 75
network configuration 69
network installation 54
root password 69
SUSE 100
updates 70
user administration 69
variants 53
Intel
CPUs 434
Spectre and Meltdown 765
Interactive shell 196
interfaces 495, 973
Internal field separator 220
Internationalization 418
Internationalized domain name 491
Internet Printing Protocol (IPP) 449
Internet Security 1039
inxi 424
ionice 304, 1032
iotop 300
ip 330
addr show 482
route 481
IP filter 1049
IP forwarding 1128
IPP 449
IP ports, list 1040
IPv6
AWS EC2 806
Debian 496
Dovecot 936
mail server 902
manual configuration 482
MySQL and MariaDB 877
nameserver 490
Postfix 913
Samba 973
SSH server 811

IPv6 (Cont.)
TCP wrapper 1047
ISO 8859 character sets 418
iso9660 file system 620
ISO image 53
iwd 487

J

j 256, 273
jed 187, 370
jmacs 187, 370
Jobs, performing regularly 318, 323
joe 188, 370
Journal (systemd) 452
journalctl 461
journald.conf 462
Journaling file systems 625
btrfs 631
ext3 627
xfs 646
jove 187, 370
jpico 188
jumpstats 256

K

kacpid 317
Kali Linux 42
kbd 396
kblockd 317
KDE 161
Neon distribution 43, 162
sharing a directory 989
kdenetwork-filesharing 989
kdevtmpfs 317
KDE Wallet 170
Kernel 38, 747
boot options 760
boot options (GRUB) 696
compiling 745, 754
configuring 750
determining the configuration 751
documentation 748
hotplug function 430
installing 754
IP filter 1049
latest version 749
logging 455
modules 736
options 741
options (GRUB) 696
Kernel (Cont.)
parameters 765
processes 316
tainted 575
update (GRUB) 694
Kernel Mode Setting 571
Key
HTTPS (Apache) 837
POP/SMTP (Dovecot) 938
SSH 814
Keyboard
blocked 303
configuration 395
GNOME 139
KDE 174
keyboard-setup 396
keyfile (NetworkManager plugin) 477, 493
keys files (GNOME MIME) 158
kgx 181
khelperd 317
kill 303
killall 303
kmod 736, 739, 741
KMS 571
Konqueror, sharing a directory 989
konsole 181
kpartx 1130
kPatch 756
KRename 169
ksh 194
ksoftirqd 317
Ksplice 756
KStatusNotifierItem (GNOME extension) 154
kswapd 317
ksysguard 301
kthread 317
Kubuntu 43, 110
KVM 1105, 1106, 1108
backup 1032
EFI 1118
SELinux 1123
kvm-ok 1106
kworker 317

L

l10n 418
LAMP/LEMP system 870
LAN 469
NetworkManager 471

LAN (Cont.)
security ... 1039
Landscape ... 507
Landscape (Ubuntu) ... 392
LANG ... 420, 421
LANGUAGE ... 421
language-selector-gnome ... 565
Language setting ... 418
Latin character sets ... 418
LC_ALL ... 421
LC_COLLATE ... 420
LC_CTYPE ... 420
LC_MESSAGES ... 420
LC_MONETARY ... 420
LC_NUMERIC ... 420
LC_PAPER ... 420
LC_TIME ... 420
LC_TYPE ... 418
ldd ... 413
Leap (openSUSE) ... 98
less ... 185
/proc files ... 759
let ... 215
Let's Encrypt (certificates) ... 844
LFS ... 627
lfs ... 273
lftp ... 344
LGPL ... 45
libcap ... 289
libdbus ... 430
libdrm ... 573
libgudev ... 430
libguestfs ... 1130, 1131
libinput ... 569
libpam-google-authenticator ... 819
libudisk2 ... 430
libvirt ... 1109
SELinux ... 1123
SSH ... 1115
libvirtd ... 1109
libwrap ... 1047
libzypp ... 521
Licenses ... 44
lightdm ... 587
lightdm.conf ... 589
Limit ... 854
Links ... 264
Linus Torvalds ... 47
Linux ... 37
compiling the kernel ... 745
configuration ... 391
distribution ... 39
Linux (Cont.)
emergence ... 44
installation ... 49, 56
kernel modules ... 736
kernel updates ... 694
Linux Standard Base ... 41
requirements ... 49
system changes ... 70
updates ... 70
linux (GRUB) ... 702
Linux Mint ... 92, 110
Linux Standard Base (LSB) ... 41
Linux Vendor Firmware Service ... 542
Live system ... 40, 55
Ubuntu ... 111
Livna package source ... 91
lm-sensors ... 437
LMTPD ... 944
ln ... 264
local_recipient_maps ... 925
locale ... 421
localectl ... 396, 419, 720
locale-gen ... 423
Locales/internationalization ... 418
localhost ... 487
Localization ... 418
localmodconfig ... 753
local networks ... 469
Local networks security ... 1039
Local variables ... 215
locate ... 267
log file ... 975
logger ... 455
Logging
Apache ... 836
logrotate ... 456
logwatch ... 457
MySQL ... 890
Postfix ... 914
Samba ... 975
Logging files ... 452
X ... 584
Logical partition ... 59
Logical volume ... 665
Logical Volume Manager ... 61, 665
backup with snapshots ... 1031
basics ... 61
RAID ... 666
server configuration ... 774
snapshots ... 669
login.defs ... 409
loginctl ... 719

Login name 405
Login shell 196
logrotate 456
Apache 836
Samba 975
logwatch 458
Loops (bash) 233
lost+found 290
lostfound 629
lp 451
lpadmin 452
lpc 451
lpd
CUPS 447
lpinfo 452
lpoptions 447, 452
lpq 451
lprm 451
lpstat 452
ls, examples 256
lsblk 424
lscpu 435
lsd 273
lsmod 738
lsof 1043
lspci 424, 427, 583, 738
lsscsi 427
lsusb 424, 427
LTS Enablement Stack 114
LTS support status (Ubuntu) 562
LTS version (Ubuntu) 109
Lubuntu 110
luksFormat 680
lvcreate 667, 1031
lvextend 668
lvremove 1031
Lynx 346
lzop 1019, 1020, 1032

M

m23 ..... 392, 507
m-a 745
MAC 1045
MAC address 330
changing (server virtualization) ... 1129
determining 482
Machine Owner Keys (Secure Boot) ... 691
macOS file system 613
magick 221
Mailbox 898, 907
Dovecot 936
Mailbox (Cont.)
mbox format 907
virtual 929
maildir format 899
Maildir mailbox
Dovecot 937
Mutt 347
Postfix 923
mailq 914
Mail → Email
Mail server 895
IPv4 902
troubleshooting 962
main.cf file (Postfix) 908, 909
Major device number 293
makepasswd 410
make-ssl-cert 843
makethumbs 221
man 189
Managed server 773
Mandatory access control (MAC) 1070
Manjaro 41
Manjaro Linux 95
Manual network configuration 491
map to guest = bad user 985
Markdown, Emacs extension 388
master.cf file (Postfix) 913
Master Boot Record 691
Matomo 862
max log size 975
mb 1035
mbox format 898
mbox mailbox 907
MBR 604, 691
partition numbers 600
md 317
md_mod (LVM) 665
md_mod (RAID) 656
MDA 896
mdadm 656, 659
mdadm.conf 657
mdnsd 315
Media server 990
Media sharing 136
medusa 812
MELPA (Emacs extensions) 388
Meltdown 542, 765
Memtest86 425
menu.lst (GRUB) 696
Mesa 572
Mesa-demo-x 583
Mesa Library 569

mesa-utils 583
Microcode update 542
Micro OS 99
MicroOS 509
Microsoft
Exchange Server 897
KVM installation 1119
SMB protocol 963
subsystem for Linux 1137
TrueType fonts 149
Windows partitions 647
migration 317
Milter
ClamAV 947
OpenDKIM 958
MIME
CUPS (print) 448
GNOME 158
KDE 175
mime.convs 448
mime.types 448
Minor device number 293
Mint 43, 92
mintbackup 94
mintdrivers 94
mintinstall 94
mintstick 94
mitigations (kernel option) 767
MIT license 45
mkconf 657
mke2fs 628
mkfs.btrfs 632
mkfs.ntfs 651
mkfs.xfs 646
mkinitramfs 693
mklabel 604
mkpasswd 410
mkswap 653
mlocate 268
Mobile Shell 825
modinfo 739
modprobe 737
modprobe.conf 479
Module, options 741
Module (DNF) 780
module-assistant 745
Modules 736
compiling 742, 754
device files 741
loading automatically 740
parameters 739
using 737
Modules (Cont.)
versioning 737
modules.alias 740
Modules (AppStream) 518
Modules (DNF) 518
MOKs (Secure Boot) 691
Monitoring the network activity 1042
Monolithic kernel 752
more 185
moreutils 204
Mosh 825
Mother program (GNOME shell) 569
mount 617
examples 618
options 620
remount for system partition 618
Mouse
blocked 303
KDE 174
MTA 895
mtab 617
Muffin (window manager) 93
Multiprocessing modules (MPM) 869
multiuser target 586
MultiViews 854
Munin 301
Music directory 158
mutt 346
mv 262
renaming files 262
security queries 91
MX entry (DNS) 902
mydestination 927
mylvmbackup 890
MySQL 873
administration 883
backups 888
IPv6 877
Workbench 885
mysql 883
mysqladmin 884
mysqldump 888

N

Nagios 301
Nameserver
client configuration 489
determining the current address 476
Ubuntu 476
Name Service Switch 416
nano 186

Nautilus
MIME ... 158
root mode ... 131
sharing a directory ... 988
nautilus-image-converter ... 137
nautilus-share ... 988
ncdu ... 259, 273
ncrack ... 812
needrestart (DPKG) ... 535
needs-restarting (DNF) ... 521
negativo package source (Fedora) ... 577
Nemo (file manager) ... 93
Neon (KDE distribution) ... 162
net ... 989
NetBIOS ... 964
Netfilter (firewall system) ... 1049
netplan ... 500
network bridge ... 1127
netstat ... 1042
net-tools ... 1042
Network ... 469
configuring the ethernet controller ... 478
network controller ... 478
security ... 1039
Network bridge ... 1126
Network configuration
manual ... 491
static ... 491
networkd ... 501
NetworkManager ... 469
network bridge ... 1126
plugins ... 477
Network status ... 329
Network Time Protocol ... 399
newaliases ... 899, 924
newgrp, example ... 282
nfs file system ... 613
nft ... 1050
example ... 1066
nftables (firewall system) ... 1049
NGINX ... 869
phpMyAdmin ... 887
nice ... 304
Night light ... 145
nmap ... 1043
nmbd ... 969
nmblookup ... 998
nmcli ... 476, 1126
nm-connection-editor ... 1126
noauto ... 621
nodeadkeys ... 395
nodev ... 621
nodev file systems ... 614
noexec ... 621
nofail ... 621
for EFI partition ... 777
nomatch ... 245
nosuid ... 621
Notebook battery ... 425
Notebook fan control ... 443
Notebook optimization ... 438
battery charging behavior ... 442
nproc ... 435
nscd ... 417, 418
NSS ... 416
ntfsclone ... 651
ntfs file system ... 647
streams ... 650
ntfsinfo ... 651
ntfslabel ... 651
ntfsprogs ... 596, 648, 651
ntfsresize ... 651
ntfsundelete ... 651
NTP ... 399
ntpd ... 399
ntpdate ... 399
nullok (PAM cfiguration) ... 820
NVIDIA ... 574
disadvantages ... 575
driver installation ... 576
Prime ... 580
Secure Boot ... 52
nvidia driver (X)
Debian ... 84
Fedora ... 92
openSUSE ... 103
Ubuntu ... 114
nvidia-prime ... 581
nvidia-settings ... 579

O

Oh My Zsh ... 250
One-Click-Install (openSUSE) ... 559
Oneshot type (systemd) ... 722
Online accounts (GNOME) ... 141
open ... 159, 298
OpenCL ... 572
OpenDKIM ... 952
OpenGL ... 571
openmediavault (OMV) ... 968
Open source ... 44
openssh ... 809

openssl 838, 919
openSUSE 42, 98
Samba 988
Snapper 642
Operating system 37
Optimizing the battery life 442
Optimus (NVIDIA) 580
Optimus hybrid graphics 575
Options 854
Apache 854
kernel 750
modules 741
Oracle
Linux 42, 782
MySQL 873
VirtualBox 1087
Oracle Cluster Filesystem (OCFS) 593
Order (Apache) 855
Origins-Patterns 534
os-prober 701
Output redirection 202
sudo 309
ovmf 1118
Owner
new files 283
of files 273
owner 621

P

p7zip 1020
paccache 540
PackageKit 540
packagekitd 541
Packages 509
Debian 524, 554
dependencies 510
management 505
package manager 557
Red Hat 509
Ubuntu 562
Packet filters 1049
Pacman 537
Manjaro 97
pacman 537
PAM 413
Google Authenticator 819
systemd 718
pam_cracklib 409
pam_faillock 411
pam_pwquality 409
pam_unix 409
Pamac 97, 540
Panel
GNOME 121
KDE 165
Parameter substitution 228
Paravirtualization 1112
parted 605
EFI partition 688
Parted Magic 43
Partition
basics 58
changing, Linux 64
file system 67
ideal partitioning 65
in the directory tree 594
naming on Linux 598
remount 618
types 59
Partition limits 605
Partitions
cloning 664
EFI 688
enlarging 804
passdb backend 977
Password 407
aging 408
changing 409
expiration date (chage) 408
forgotten 410
for groups 411
quality 409
root 409
Password management
Apache 857
PAM 413
Samba 977
PATH 216
path 983
Pattern (ZYpp) 524
pavucontrol 434
PCI-Bus 427
pdbedit 978
pdksh 194
Periodic execution of jobs 318, 323
pesign 749
PGP 901
Phonon 433
PHP 865
Apache MPM 869
Emacs extension 388
Unicode 834
phpMyAdmin 886

Physical extent 665
Physical volume 665
PID 300
PID file 301
pidof 301
ping 331
Pipes 204
Piwik 862
pkcon 541
pkexec 306, 313
pkmon 541
Plasma 162
Plesk Panel 392
plocate 268
Pluggable Authentication
  Modules (PAMs) 413
Plugins, DNF 516
pnuke 825
policycoreutils-gui 1073
Policy files (X) 313
PolicyKit 312
Pop!_OS 43, 104
  systemd-boot 708
POP server 931
  authentication 937
Portland Project 158
Port number
  FTP (20, 21) 1040
  HTTP (80) 1040
  IMAP (25, 587) 898
  list 1040
  reference 1040
  SMTP (25, 587) 898
Ports (TCP/IP) list 1040
Port scan 1043
Postfix 906
  alias 924
  IPv6 913
  logging 914
  virtual domains 927
postmap 909, 926
postqueue 914
PostScript 445
  printer definition (PPD) 448
power-profiles-daemon 437
Power saving functions 425
powertop 438
PPAs (Ubuntu) 562
PPD files 448
ppds.dat 448
prefork module (Apache) 869
pri (swap priority) 652
Primary Domain Controller 967
Primary partition 59
PRIME (NVIDIA graphics) 580
prime-select 581
Print
  MIME (CUPS) 448
  PostScript 445
printcap
  CUPS 447
printenv 215
printers.conf 447
Printer server 444
Printing 444
  automatic data conversion 446
  configuration 444
  daemon (CUPS) 447
  filters 446
  GNOME 141
  KDE 174
  managing print jobs 451
  spooling system 445
  using commands 451
pro 563
Processes 297
  background processes 298
  compute time 304
  foreground processes 298
  hierarchy 302
  limiting the size 304
  managing 299
  priority 304
  running regularly 318, 323
  running under a different
    identity 305
  terminating by force 303
Procmail 896
profile 196
profile files 215, 216
Program 297
  launching 298
  see also Processes 297
  starting (bash) 206
Prompt
  bash 196
  zsh 251
PROMPT_COMMAND (variable) 197
Proxmox 1114
Proxy configuration 474
ps 299
PS1 (variable) 196
pssh 825
pstree 302

Public directory 158
pw-top 434

Q

QCOW2 format
enlarging an image 1134
QEMU 1105, 1106
qemu-img 1130, 1134
qemu-kvm 1108
qemu-system-x86 1108
Qt 161

R

radeon 573
RAID 654
btrfs 638
LVM 666
monitoring 657
scrubbing 662
server configuration 774
RANDOM_DELAY variable 323
RandR 572
rb 1035
rc.local 730
rc files 727
rdiff-backup 1025
RDP, Wayland 570
read 230
readline 196
Reboot
forcing 726
required after update 521, 535
reboot 725
Rebooting the host system 1111
reboot-required file 535
recover-file (Emacs) 371
recycle 987
Recycle bin (Samba) 986
Red Hat 42, 779
initrd file 693
Satellite 507
static network configuration 492
sudo 311
Red Hat Enterprise Linux (RHEL) 779
redshift program 145
Regular execution of jobs 318, 323
Regular expressions, Emacs 380
reload (Init-V process) 728
RemainAfterExit keyword
(systemd) 722
Remi package source 560, 867
Remmina 147
Remote maintenance 147
remount (system partition) 618
remove 742
Rendezvous 502
renice 304
Require (Apache) 855
reset 185
resize2fs 630
resolv.conf 489
Ubuntu 490
resolvectl 476
Restart, forcing 726
restart (init-V process) 728
Restarting the host system 1111
restorecon 1072, 1076
Retina monitors
GNOME 144
KDE 172
Retina screens, VirtualBox 1101
Reverse DNS 905
AWS EC2 806
RHEL
firewall 1051
keyboard 396
system startup 729
RHSM 779
Richard Stallman 46
RIDL 765
rlogin 334
rm 258
security queries 91
rmmod 739
Rocky Linux 783
Rolling release 99
Ubuntu kernel 114
root 69, 409
Root partition 65
Root server 772
route 480, 481
RPM 509
rpm
cannot open packages database 511
examples 511
repairing the database 511
RPM Fusion 91
RPMS 510
rsnapshot 1027
rsync 1022
rsyslog.conf 452
rsyslogd 452

run-crons 322
Runlevel 727
run-parts 321
rygel 136

S

S/MIME 901
S3 cloud service 1032
Samba 963
  /etc/fstab 1003
  commissioning 969
  Fedora 988
  firewall 972
  guests 985
  IPv6 973
  Nautilus 133
  passwords 977
  recycle bin 986
  RHEL 988
  security mechanisms 965
  SELinux 974
  setting up network directories 982
  SUSE 988
  Ubuntu 988
Sample bash scripts 217
Sandbox (Flatpak/Snap) 548
scp 338
screen 826
Screen sharing 147
Screenshots, Wayland 570
Script
  bash 217, 225
  programming 217
ScriptAlias 853
Scripts
  SSH 337
Scrubbing (RAID) 662
SCSI 598
scsi_eh 317
SD card, formatting 595
seahorse 816
seahorse-nautilus 137
sealert 1076
search (GRUB) 702
Searching, Emacs 379
Secure Boot 51, 690
  NVIDIA 52
Secure Shell 809
Secure Sockets Layer 836
Security 1039
security (Samba) 971
Security context 1071
securityfs 1078
Security token 822
sed example 262
Selector (syslog) 453
SELinux 1069, 1070
  Apache 831
  libvirt 1123
  opendkim 954
  Samba 974
  SSH 817
  SSH port 811
Sender Policy Framework (SPF) 950
sensors 437
sensors-detect 437
Server
  database (MySQL) 873
  Samba 963
  SSH 809
  web server (Apache) 829
  X 568
ServerAlias 860
Server installation 771
Server Message Block 963
ServerName 860
ServerName (Apache) 834
server role (Samba) 971
server string 970
service 721, 728
Services 315
set 210, 215
setcap 289
setenforce 1076
setfacl 286
setfattr 288
Setgid bit 280
setopt 246
setroubleshoot-server 1076
setsebool 1073
Setting up a hotspot (Wi-Fi) 473
Setting up users 402
Setuid bit 279
setup.exe 509
sfdisk 664
sftp 342
  server 810
sgdisk 664
shadow 407
Shared folder (VirtualBox) 1099
Share-level security 965
Shares 965
  GNOME settings 989

Shares (Cont.)
media (GNOME) ... 136
Nautilus ... 134, 988
WebDAV (GNOME) ... 136
shebang ... 218
Shell
bash ... 193
changing the default shell ... 194
variables ... 213, 225
zsh ... 241
Shell programming ... 217
examples ... 217, 325
Shim ... 52, 690
shopt ... 210
Shutdown
forcing ... 726
logging ... 724
of the host system ... 1111
shutdown ... 725
Sieve ... 944
Silverblue ... 85, 509
Silverblue (Fedora) ... 85
Simple Storage Service (S3) ... 1032
Single-user mode, systemd ... 716
Slax ... 43
Sleep mode ... 426
smartctl ... 670
smartd ... 674
smb.conf ... 969
smbclient ... 999
smbd ... 969
smbfs file system ... 1000
SMB log ... 963
smbpasswd ... 978
SMB protocol, disabling version 1 ... 973
smbstatus ... 969, 972
smbtree ... 998
SMB versions ... 965
SMT ... 767
smtp_tls parameter (Postfix) ... 916
SMTP, troubleshooting ... 962
smtpd_tls parameter (Postfix) ... 916
Snake-oil certificate and key
Apache ... 843
Postfix ... 915
Snap ... 550
snapd ... 551
Snapper (openSUSE) ... 642
Snapshots
btrfs ... 636
LVM ... 669
socat ... 1122
Socket files ... 264
soft_bounce (Postfix) ... 913
Software installation ... 505
software-properties ... 564
software-properties-gtx ... 529
Software RAID ... 654
source ... 225
Source package ... 510
sources.list ... 528
SpamAssassin ... 941
Spam protection ... 941
speaker-test ... 433
Special bits (access rights) ... 279
Special characters
bash ... 237
Spectre ... 542, 765
spectre-meltdown-checker.sh ... 766
spelling
bash script ... 219
SPF entry (mail server) ... 950
Spice ... 1113
spice-vdagent ... 1118
Spin (Fedora) ... 85
splash ... 760
sponge ... 204
Spooling system (printing) ... 445
squashfs file system
Snap ... 551
SRPMS ... 510
SSD TRIM ... 675
SSH ... 809
changing the port ... 811
Google Authenticator ... 818
IPv6 ... 811
libvirt ... 1115
Mobile Shell ... 825
securing ... 810
SELinux ... 817
ssh ... 334
avoiding the login ... 814
file system ... 340
Konqueror ... 170
port redirection ... 450
server ... 809
tunnel ... 339
ssh-agent ... 816
sshd ... 809
ssh-keygen ... 814
SSL ... 836
SSL (Apache) ... 836
ssl-cert-snakeoil.key ... 843
ssl-cert-snakeoil.pem ... 843

SSLCipherSuite (Apache) 849
SSLEngine (Apache) 842
SSLProtocol (Apache) 849
SSLStrictSNIVHostCheck (Apache) 859
SSLxxxFile (Apache) 842
Stable packages 555
Stallman, Richard 44
Standard input 202
Standard output 202
star 288
STARTTLS
  Dovecot 937
Static network configuration 491
Sticky bit 281, 283
Streams (NTFS file system) 650
Stretch 77
Strings
  bash 213
  parameter substitution (bash) 228
stripcomments (bash example) 219
su 306
  graphic variant 306
  Wayland 570
Subiquity 791
submission (Postfix) 913
Subsequent installation 70
Substitution mechanisms 208
Subvolumes (btrfs) 635
sudo 307
  Fedora 311
  input/output redirection 309
  Ubuntu 309, 311
  Wayland 309, 570
suid 279
Sun, ZFS file system 612
SUSE 98
  AppArmor 1077
  firewall 1055
  kernel configuration 751
  package management 557
  Samba 988
  Snapper 642
Suspend mode
  forcing a restart 726
Suspend to Disk 426
swaks 943
Swap file 66, 653
swapon 653
Swap partition 66
  EC2 instance 800
  mounting 651
swappiness parameter 652
Symbolic links 265
  versus bind mounts 623
Synaptic 535
sync (S3) 1035
Synchronization tools 1013
Syntax highlighting 385
sysctl 765
syslog 975
system76-power 108
System Administration 391
system-config-printer 142
system-config-samba 989
system-config-selinux 1073
systemctl 323, 713
systemd 711
  as Cron replacement 323
  custom service file 721
  Fedora 729
  network configuration 501
  network device names 491
  RHEL 729
  running processes periodically 323
  starting the graphics system 586
  timers 323
  WSL 1140
systemd-boot 707
systemd-gpt-auto-generator 624
systemd-journald 460
systemd-networkd 501
systemd-resolved 476
systemd-timedated 398
systemd-timesyncd 400
systemd-udev 294
systemd-udevd 294
systemd-vconsole-setup 397
System partition 65
  remount 618
System Security Services Daemon 418
System settings, KDE 170
System startup
  GRUB 687
  init-V 726
System-V init process 726

T

Tabs (Emacs) 376
tail 185
Tainted kernel 575
tar 1019, 1021
targeted 1073
tasksel 532

TCP wrapper library 1046
tcsh 194
TDB 978
TeamViewer 148
tee 205
telnet 334, 340
  SMTP troubleshooting 962
Temperature (CPU) 437
Templates directory 158
Terminal 181
test 231
Tethering 473
Text console 179
  configuration 395
  font 397
  keyboard 395
Text editors 186, 369
Text file, searching 270
Themes (GNOME) 156
Themes (KDE) 173
thinkfan 443
Third mouse button 128
Thumbnails 221
Thunderbolt 431
tigervnc viewer 147
Tilde 254
Tiling Manager for GNOME 154
Time 397
Time-controlled execution
  of jobs 318, 323
timedatectl 720
Timers (systemd) 323
timeshift 94
timestamp_timeout 307
time-sync 399
Time zone 397
  glibc 398
TinyCore 43
tldr 273
TLP 440
tlp-stat 441
TLS 901
  Dovecot 937
Token 822
Torvalds, Linus 47
traceroute 332
Transport Layer Security 901
Trash can (GNOME) 132
TRIM
  LUKS encryption 681
TRIM (SSDs) 675
Troll Tech 161
Trusted Platform Module (TPM) 1119
Trusted TLS Connection (Postfix) 917
ttf-mscorefonts-installer 149
Tumbleweed 99
tune2fs 630
tune2fs -l 629
Tunnel, SSH 339
Turbo boost 436
TurboPrint 445
Two-factor authentication (2FA) 818
Type keyword (systemd) 722
type name 199

U

Ubuntu 43
  initrd file 693
  keyboard 395
  package management 562
  Pro 563, 790
  static network configuration 499
  sudo 309, 311
  system startup 730
  VirtualBox 1096
ubuntu-drivers 114, 564
Ubuntu Pro 756
ubuntu-restricted-extras 114
Ubuntu Server 110
Ubuntu Studio 110
udev 430
udev system 294
  network device names 491
udisks2 430
UDP 1041
UEFI 50
  in KVM/QEMU 1118
  partition 51
  Secure Boot 51
UEFI Secure Boot 690
ufw 1055
UID 405
ulimit 304
umask 283
unattended-upgrades 533
Unicode 418
  Apache 834
  console 397
  file system 253
  PHP 834
Univention Corporate Server 392
Unix 37
unset 215

Unstable packages 555
Untrusted TLS Connection
(Postfix) 917
unxz 1019
unzip 1019
update-alternatives 546
updatedb 268
update-grub 696, 698
update-initramfs 693
update-ms-fonts 149
Updates 70
BIOS/EFI/firmware 541
upower 430
Upstart, Ubuntu 730
USB 427
USB flash drive, formatting 595
usbreset 427
user_xattr 285
useradd 403
User-level security 966
username map 980
Users 404
groups 274
manging 401
setting up 402
usershare allow guests 986
User shares (Samba) 986
User themes 156
UTC (Universal Time, Coordinated) ... 397
UTF-8 418
UUID
in /dev/disk 602
setting (ext3) 630
setting (xfs) 647

V

valid users 983
Vanilla kernel 750
Vanilla OS 509
Variables
bash 213, 225, 230
vboxdrv 1088
vboxguest 1096
vboxmanage 1102, 1103
vboxnetadp 1088
vboxnetflt 1088
vboxsf file system 1100
VDPAU 572
vfat file system 647
vgcreate 667
vgscan 667
Vi 187, 351
Videos directory 158
Vim 187, 351
character set 365
configuration 364
cursor movement 355
easy mode 367
editing multiple files 361
editing text 356
macros 366
mouse 366
options 363
search and replace 360
swap file 365
tabs 366
Unicode 365
vimrc file 364
Vinagre 147
virsh 1108, 1119
virt-cat 1133
virt-clone 1123
virt-df 1131
virt-edit 1133
virt-filesystems 1132
virt-inspector 1132
virtio driver 600
virtio drivers 1112
virt-make-fs 1133
virt-manager 1108, 1113
virt-resize 1134
virt-sparsify 1135
virt-sysprep 1123
virt-tar-in 1133
virt-tar-out 1133
virt-top 1125
virtual_alias_domains 928
VirtualBox 1087
converting an image to QCOW2 format 1134
Virtual domains (Postfix) 927
Virtual file systems 614
Virtual hosts 859
with HTTPS 859
Virtual mailboxes 929
Virtual private cloud (VPC) 806
Virtual private networks (VPN) 473
virt-viewer 1124
virt-viewer vmname 1124
Virus protection 947
VISUAL 188
visudo 308
vmlinuz 755

vmlinuz file 689
VNC 147
Wayland 570
Volume group 665
Volumes
LVM 62
Vorta 1019
VRFY command (Postfix) 931
vulnerabilities files 766

W

w3m 346
WannaCry malware 973
watchdog 317
Wayland 567
limitations 570
log 585
sudo 309
WSL 1137
Webalizer 861
Web browser, text mode 346
WebDAV, GNOME sharing 136
Web directory, securing 857
Web hosting 773
Webmin 392
Web server 829
website 1035
wget 343
whereis 199, 267
which 266
while (bash) 235
Whitelist (SpamAssassin) 944
Wi-Fi
access point 473
NetworkManager 472
Wildcard 259
Wildcard certificate 847
Wildcard characters 209
complications 261
Window buttons
GNOME 150
Window manager 568
Windows
file system 647
hibernate 648
KVM installation 1119
network directories 963, 1000
subsystem for Linux 1137
Windows Subsystem for
Linux (WSL) 1137
network integration 1145
WINS 964
WordPress 891
workgroup 970
Workgroup security 966
Workspace
KDE 166
WPA 485
wpa_passphrase 486
WPA2 485
wpasupplicant 485
in /etc/network/interfaces 497
writeable 983
wsdd 976
WS-Discovery 976
WS-Discovery (WSD) 964
wsl.conf 1140, 1148
WSL2 1145
WSL command 1146
wslconfig 1147
WSLg 1143

X

X 567
determining the version 582
log 584
logging 584
server 568
SSH 338
window manager 568
X11R6 568
xargs 212
XDG 158
xdg-desktop-icon 159
xdg-desktop-menu 159
xdg-desktop-portal 571
xdg-email 159
xdg-icon-resource 159
xdg-mime 159
xdg-open 159
xdg-open filename 298
xdg-screensaver 159
xdg-user-dirs 158
xdg-user-dirs-gtk 158
xdm 587
xdpyinfo 583
XE 569
xfs_admin 647
xfs_check 647
xfs_repair 647
xfs file system 646
xkill 303

Xorg.0.log 584
XRender 572
xsensors 437
Xubuntu 43, 110
xvda devices 800
Xwayland 569
X window system 567
xz 1019, 1020

Y

YaST 98
package management 557
yay 539
YOU 557
YubiKey 822
Yum
automatic updates 521
package manager 513
yum.conf 514

Z

z 273
Zentyal 392
ZENworks 392
Zeroconf 502
Zero Install 509
ZFS file system 612
zile 370
zip 1019
zipinfo 1019
ZombieLoad 765
Zorin OS 110
zsh 241
zsh-newuser-install 244
zshrc 243
zsh-syntax-highlighting 252
ZYpp 521
zypper 522

Service Pages

The following sections contain notes on how you can contact us.

Praise and Criticism

We hope that you enjoyed reading this book. If it met your expectations, please do recommend it. If you think there is room for improvement, please get in touch with the editor of the book: meaganw@rheinwerk-publishing.com. We welcome every suggestion for improvement but, of course, also any praise!

You can also share your reading experience via Twitter, Facebook, or email.

Technical Issues

If you experience technical issues with your e-book or e-book account at Rheinwerk Computing, please feel free to contact our reader service: support@rheinwerk-publishing.com.

About Us and Our Program

The website http://www.rheinwerk-computing.com provides detailed and first-hand information on our current publishing program. Here, you can also easily order all of our books and e-books. Information on Rheinwerk Publishing Inc. and additional contact options can also be found at http://www.rheinwerk-computing.com.

Legal Notes

This section contains the detailed and legally binding usage conditions for this e-book.

Copyright Note

This publication is protected by copyright in its entirety. All usage and exploitation rights are reserved by the author and Rheinwerk Publishing; in particular the right of reproduction and the right of distribution, be it in printed or electronic form.

© 2024 by Rheinwerk Publishing, Inc., Boston (MA)

Your Rights as a User

You are entitled to use this e-book for personal purposes only. In particular, you may print the e-book for personal use or copy it as long as you store this copy on a device that is solely and personally used by yourself. You are not entitled to any other usage or exploitation.

In particular, it is not permitted to forward electronic or printed copies to third parties. Furthermore, it is not permitted to distribute the e-book on the Internet, in intranets, or in any other way or make it available to third parties. Any public exhibition, other publication, or any reproduction of the e-book beyond personal use are expressly prohibited. The aforementioned does not only apply to the e-book in its entirety but also to parts thereof (e.g., charts, pictures, tables, sections of text).

Copyright notes, brands, and other legal reservations as well as the digital watermark may not be removed from the e-book.

Digital Watermark

This e-book copy contains a **digital watermark**, a signature that indicates which person may use this copy. If you, dear reader, are not this person, you are violating the copyright. So please refrain from using this e-book and inform us about this violation. A brief email to info@rheinwerk-publishing.com is sufficient. Thank you!

Trademarks

The common names, trade names, descriptions of goods, and so on used in this publication may be trademarks without special identification and subject to legal regulations as such.

All products mentioned in this book are registered or unregistered trademarks of their respective companies.

Limitation of Liability

Regardless of the care that has been taken in creating texts, figures, and programs, neither the publisher nor the author, editor, or translator assume any legal responsibility or any liability for possible errors and their consequences.