FOURTH EDITION

"As an in-depth guide to Ubuntu, this book does an exceptional job of covering everything from using the Unity desktop to exploring key server configurations"

—**Elizabeth K. Joseph,** Systems Administrator & Ubuntu Community Member

A Practical Guide to Ubuntu Linux®

- **The #1 Ubuntu resource**, fully updated for Ubuntu 14.04 (Trusty Tahr)—the Long Term Support (LTS) release Canonical will support into 2019
- **Extensive new coverage** of installation, security, virtualization, MariaDB, Python Programming, and much more
- **Updated JumpStarts** help you set up many complex servers in minutes
- **Hundreds of up-to-date examples,** plus four updated indexes deliver fast access to reliable answers

INCLUDES
FULL UBUNTU ON DVD

Mark G. Sobell

FREE SAMPLE CHAPTER

SHARE WITH OTHERS

Praise for Books by Mark G. Sobell

"I have said before on several occasions that Sobell does really good work. Well, [*A Practical Guide to Ubuntu Linux®, Third Edition*] holds true to my words. This is a big book with some 1250+ pages in it absolutely filled to the brim with useful information. The review on the front cover mentions that the book is 'comprehensive' and that just might be understating it a little. This book has practically anything you might want to know about Ubuntu, and references a lot of really helpful general Linux and userland program information and it's put together in a very straight forward and understandable way. Having the word 'Practical' in the name is also a really good fit as the book offers great walk-throughs on things people will want to do with their Ubuntu install from beginner things like configuring a printer all the way up to things like some Perl programming and running your own Web server. All in all, this book is not only worth a look, but it is a keeper. It's a good read and great technical reference."

—Lincoln C. Fessenden, Linux Guy / I.T. Manager

"The third updated edition of *A Practical Guide to Ubuntu Linux®* offers a fine reference perfect for any Ubuntu Linux computer collection, packing in hundreds of practical applications for Ubuntu with keys to security, Perl scripting, common administration tasks, and more. From keeping Ubuntu systems current to handling configuration issues, this is a solid reference to the latest Ubuntu applications and challenges."

—Jim Cox, Midwest Book Review

"This is an excellent text and I am using it as of this term as the textbook for the class in Linux that I am teaching at the local Community College. The first book on UNIX that I used twenty-five years ago was written by Sobell. He hasn't lost his touch."

—James J. Sherin, Part-Time Faculty, Westmoreland County Community College

"When I first started working with Linux just a short 10 years or so ago, it was a little more difficult than now to get going. . . . Now, someone new to the community has a vast array of resources available on the web, or if they are inclined to begin with Ubuntu, they can literally find almost every single thing they will need in the single volume of Mark Sobell's *A Practical Guide to Ubuntu Linux®*.

"Overall, I think it's a great, comprehensive Ubuntu book that'll be a valuable resource for people of all technical levels."

—John Dong, Ubuntu Forum Council Member, Backports Team Leader

"I would so love to be able to use this book to teach a class about not just Ubuntu or Linux but about computers in general. It is thorough and well written with good illustrations that explain important concepts for computer usage."

—Nathan Eckenrode, New York Local Community Team

“Ubuntu is gaining popularity at the rate alcohol did during Prohibition, and it’s great to see a well-known author write a book on the latest and greatest version. Not only does it contain Ubuntu-specific information, but it also touches on general computer-related topics, which will help the average computer user to better understand what’s going on in the background. Great work, Mark!”

—Daniel R. Arfsten, Pro/ENGINEER Drafter/Designer

“This is well-written, clear, comprehensive information for the Linux user of any type, whether trying Ubuntu on for the first time and wanting to know a little about it, or using the book as a very good reference when doing something more complicated like setting up a server. This book’s value goes well beyond its purchase price and it’ll make a great addition to the Linux section of your bookshelf.”

—Linc Fessenden, Host of The LinuxLink TechShow, tllts.org

“Overall, *A Practical Guide to Ubuntu Linux®* by Mark G. Sobell provides all of the information a beginner to intermediate user of Linux would need to be productive. The inclusion of the Live DVD of Ubuntu makes it easy for the user to test-drive Linux without affecting his installed OS. I have no doubts that you will consider this book money well spent.”

—Ray Lodato, Slashdot contributor, www.slashdot.org

“I’m sure this sounds a bit like hyperbole. Everything a person would need to know? Obviously not everything, but this book, weighing in at just under 1200 pages, covers so much so thoroughly that there won’t be much left out. From install to admin, networking, security, shell scripting, package management, and a host of other topics, it is all there. GUI and command line tools are covered. There is not really any wasted space or fluff, just a huge amount of information. There are screen shots when appropriate but they do not take up an inordinate amount of space. This book is information-dense.”

—JR Peck, Editor, GeekBook.org

“Sobell tackles a massive subject, the vast details of a Linux operating system, and manages to keep the material clear, interesting and engaging. . . . If you want to know how to get the most out of your Red Hat, Fedora, or CentOS system, then this is one of the best texts available, in my opinion.”

—Jesse Smith, Feature Writer for DistroWatch

“I had the chance to use your UNIX books when I when was in college years ago at Cal Poly, San Luis Obispo, CA. I have to say that your books are among the best! They’re quality books that teach the theoretical aspects and applications of the operating system.”

—Benton Chan, IS Engineer

“I currently own one of your books, *A Practical Guide to Linux®*. I believe this book is one of the most comprehensive and, as the title says, practical guides to Linux I have ever read. I consider myself a novice and I come back to this book over and over again."

—Albert J. Nguyen

"The book has more than lived up to my expectations from the many reviews I read, even though it targets FC2. I have found something very rare with your book: It doesn't read like the standard technical text, it reads more like a story. It's a pleasure to read and hard to put down. Did I say that?! :-)"

—David Hopkins, Business Process Architect

"Thanks for your work and for the book you wrote. There are really few books that can help people to become more efficient administrators of different workstations. We hope (in Russia) that you will continue bringing us a new level of understanding of Linux/UNIX systems."

—Anton Petukhov

"Mark Sobell has written a book as approachable as it is authoritative."

—Jeffrey Bianchine, Advocate, Author, Journalist

"Since I'm in an educational environment, I found the content of Sobell's book to be right on target and very helpful for anyone managing Linux in the enterprise. His style of writing is very clear. He builds up to the chapter exercises, which I find to be relevant to real-world scenarios a user or admin would encounter. An IT/IS student would find this book a valuable complement to their education. The vast amount of information is extremely well balanced and Sobell manages to present the content without complicated asides and meandering prose. This is a 'must have' for anyone managing Linux systems in a networked environment or anyone running a Linux server. I would also highly recommend it to an experienced computer user who is moving to the Linux platform."

—Mary Norbury, IT Director, Barbara Davis Center, University of Colorado at Denver, from a review posted on slashdot.org

"Excellent reference book, well suited for the sysadmin of a Linux cluster, or the owner of a PC contemplating installing a recent stable Linux. Don't be put off by the daunting heft of the book. Sobell has striven to be as inclusive as possible, in trying to anticipate your system administration needs."

—Wes Boudville, Inventor

"The JumpStart sections really offer a quick way to get things up and running, allowing you to dig into the details of the book later."

—Scott Mann, Aztek Networks

"*A Practical Guide to Red Hat® Linux®* is a brilliant book. Thank you Mark Sobell."

—C. Pozrikidis, University of California at San Diego

“Overall I found this book to be quite excellent, and it has earned a spot on the very front of my bookshelf. It covers the real ‘guts’ of Linux—the command line and its utilities—and does so very well. Its strongest points are the outstanding use of examples, and the Command Reference section. Highly recommended for Linux users of all skill levels. Well done to Mark Sobell and Prentice Hall for this outstanding book!”

—Dan Clough, Electronics Engineer and Slackware Linux User

“This book presents the best overview of the Linux operating system that I have found. . . . [It] should be very helpful and understandable no matter what the reader’s background: traditional UNIX user, new Linux devotee, or even Windows user. Each topic is presented in a clear, complete fashion and very few assumptions are made about what the reader knows. . . . The book is extremely useful as a reference, as it contains a 70-page glossary of terms and is very well indexed. It is organized in such a way that the reader can focus on simple tasks without having to wade through more advanced topics until they are ready.”

—Cam Marshall, Marshall Information Service LLC, Member of Front Range UNIX, Users Group [FRUUG], Boulder, Colorado

“Conclusively, this is THE book to get if you are a new Linux user and you just got into RH/Fedora world. There’s no other book that discusses so many different topics and in such depth.”

—Eugenia Loli-Queru, Editor in Chief, OSNews.com

“This book is a very useful tool for anyone who wants to ‘look under the hood’ so to speak, and really start putting the power of Linux to work. What I find particularly frustrating about man pages is that they never include examples. Sobell, on the other hand, outlines very clearly what the command does and then gives several common, easy-to-understand examples that make it a breeze to start shell programming on one’s own. As with Sobell’s other works, this is simple, straight-forward, and easy to read. It’s a great book and will stay on the shelf at easy arm’s reach for a long time.”

—Ray Bartlett, Travel Writer

“Totally unlike most Linux books, this book avoids discussing everything via GUI and jumps right into making the power of the command line your friend.”

—Bjorn Tipling, Software Engineer, ask.com

“This book is the best distro-agnostic, foundational Linux reference I’ve ever seen, out of dozens of Linux-related books I’ve read. Finding this book was a real stroke of luck. If you want to really understand how to get things done at the command line, where the power and flexibility of free UNIX-like OSes really live, this book is among the best tools you’ll find toward that end.”

—Chad Perrin, Writer, TechRepublic

"Thank you for writing a book to help me get away from Windows XP and to never touch Windows Vista. The book is great; I am learning a lot of new concepts and commands. Linux is definitely getting easier to use."

—James Moritz

"I am so impressed by how Mark Sobell can approach a complex topic in such an understandable manner. His command examples are especially useful in providing a novice (or even an advanced) administrator with a cookbook on how to accomplish real-world tasks on Linux. He is truly an inspired technical writer!"

—George Vish II, Senior Education Consultant, Hewlett-Packard Company

"I read a lot of Linux technical information every day, but I'm rarely impressed by tech books. I usually prefer online information sources instead. Mark Sobell's books are a notable exception. They're clearly written, technically accurate, comprehensive, and actually enjoyable to read."

—Matthew Miller, Senior Systems Analyst/Administrator, BU Linux Project, Boston University Office of Information Technology

"The author has done a very good job at clarifying such a detail-oriented operating system. I have extensive Unix and Windows experience and this text does an excellent job at bridging the gaps between Linux, Windows, and Unix. I highly recommend this book to both 'newbs' and experienced users. Great job!"

—Mark Polczynski, Information Technology Consultant

"I have been wanting to make the jump to Linux but did not have the guts to do so—until I saw your familiarly titled *A Practical Guide to Red Hat® Linux®* at the bookstore. I picked up a copy and am eagerly looking forward to regaining my freedom."

—Carmine Stoffo, Machine and Process Designer to the pharmaceutical industry

"I am currently reading *A Practical Guide to Red Hat® Linux®* and am finally understanding the true power of the command line. I am new to Linux and your book is a treasure."

—Juan Gonzalez

This page intentionally left blank

A Practical Guide to Ubuntu Linux®

FOURTH EDITION

This page intentionally left blank

A Practical Guide to Ubuntu Linux®

FOURTH EDITION

Mark G. Sobell

PRENTICE
HALL

Upper Saddle River, NJ • Boston • Indianapolis • San Francisco
New York • Toronto • Montreal • London • Munich • Paris • Madrid
Capetown • Sydney • Tokyo • Singapore • Mexico City

Many of the designations used by manufacturers and sellers to distinguish their products are claimed as trademarks. Where those designations appear in this book, and the publisher was aware of a trademark claim, the designations have been printed with initial capital letters or in all capitals.

The author and publisher have taken care in the preparation of this book, but make no expressed or implied warranty of any kind and assume no responsibility for errors or omissions. No liability is assumed for incidental or consequential damages in connection with or arising out of the use of the information or programs contained herein.

For information about buying this title in bulk quantities, or for special sales opportunities (which may include electronic versions; custom cover designs; and content particular to your business, training goals, marketing focus, or branding interests), please contact our corporate sales department at corpsales@pearsoned.com or (800) 382-3419.

For government sales inquiries, please contact governmentsales@pearsoned.com.

For questions about sales outside the United States, please contact international@pearsoned.com.

Visit us on the Web: informit.com/ph

Library of Congress Cataloging-in-Publication Data

Sobell, Mark G.
A practical guide to Ubuntu Linux / Mark G. Sobell.—Fourth edition.
pages cm
Includes index.
ISBN 978-0-13-392731-3 (pbk. : alk. paper)
1. Ubuntu (Electronic resource) 2. Linux. 3. Operating systems (Computers) I. Title.
QA76.76.O63S59497 2015
005.4'46—dc23
2014036709

Copyright © 2015 Mark G. Sobell

All rights reserved. Printed in the United States of America. This publication is protected by copyright, and permission must be obtained from the publisher prior to any prohibited reproduction, storage in a retrieval system, or transmission in any form or by any means, electronic, mechanical, photocopying, recording, or likewise. To obtain permission to use material from this work, please submit a written request to Pearson Education, Inc., Permissions Department, One Lake Street, Upper Saddle River, New Jersey 07458, or you may fax your request to (201) 236-3290.

ISBN-13: 978-0-13-392731-3
ISBN-10: 0-13-392731-8

Text printed in the United States on recycled paper at Edwards Brothers Malloy in Ann Arbor, Michigan.

First printing, December 2014

For my father,
Morton Sobell

This page intentionally left blank

Brief Contents

List of JumpStarts xv
Contents xvii
Preface xliii

1 Welcome to Linux 1

PART I Installing Ubuntu Linux 23

2 Installation Overview 25
3 Step-by-Step Installation 55

PART II Using Ubuntu Linux 95

4 Introduction to Ubuntu 97
5 The Shell 149
6 The Linux Filesystem 183
7 The Linux Utilities 223
8 Networking and the Internet 285

PART III System Administration 331

9 The Bourne Again Shell (bash) 333
10 System Administration: Core Concepts 425
11 Files, Directories, and Filesystems 479
12 Finding, Downloading, and Installing Software 509

13 Printing with CUPS 539
14 Administration Tasks 563
15 System Security 595
16 Configuring and Monitoring a LAN 661
17 Setting Up Virtual Machines Locally and in the Cloud 687

PART IV Using Clients and Setting Up Servers 711

18 The OpenSSH Secure Communication Utilities 713
19 The rsync Secure Copy Utility 741
20 FTP: Transferring Files Across a Network 753
21 postfix: Setting Up Mail Servers, Clients, and More 779
22 NIS and LDAP 813
23 NFS: Sharing Directory Hierarchies 843
24 Samba: Linux and Windows File and Printer Sharing 869
25 DNS/BIND: Tracking Domain Names and Addresses 891
26 ufw, gufw, and iptables: Setting Up a Firewall 923
27 Apache (apache2): Setting Up a Web Server 951

PART V Programming Tools 1001

28 Programming the Bourne Again Shell (bash) 1003
29 The Python Programming Language 1103
30 The MariaDB SQL Database Management System 1135

PART VI Appendixes 1159

A Regular Expressions 1161
B Help 1171
C Keeping the System Up to Date Using yum 1177
D LPI and Comptia Certification 1183

Glossary 1231
JumpStart Index 1285
File Tree Index 1287
Utility Index 1291
Main Index 1297

JumpStarts

JumpStarts get you off to a quick start when you need to use a client or set up a server. Once you have the client or server up and running, you can refine its configuration using the information presented in the sections following each JumpStart.

APT (Software Packages)

JumpStart: Installing and Removing Software Packages Using apt-get 512

CUPS (Printing)

JumpStart I: Configuring a Printer Using system-config-printer 542
JumpStart II: Setting Up a Local or Remote Printer 544

OpenSSH (Secure Communication)

JumpStart I: Using ssh and scp to Connect to an OpenSSH Server 716
JumpStart II: Starting an OpenSSH Server 728

FTP (Download and Upload Files)

JumpStart I: Downloading Files Using ftp 756
JumpStart II: Starting a **vsftpd** FTP Server 765

Email

JumpStart: Configuring **postfix** to Use Gmail as a Smarthost 787

NFS (Network Filesystem)

JumpStart I: Mounting a Remote Directory Hierarchy 848
JumpStart II: Configuring an NFS Server Using shares-admin 855

Samba (Linux/Windows File Sharing)

JumpStart: Configuring a Samba Server Using shares-admin 879

DNS (Domain Name Service)

JumpStart: Setting Up a DNS Cache 906

Apache (HTTP)

JumpStart: Getting Apache Up and Running 955

Contents

Preface xliii

Chapter 1: Welcome to Linux 1

The History of UNIX and GNU–Linux 2
- The Heritage of Linux: UNIX 2
- Fade to 1983 3
- Next Scene, 1991 4
- The Code Is Free 5
- Linux Is More Than a Kernel 6
- Open-Source Software and Licensing 6
- What Is So Good About Linux? 7

Overview of Linux 11
- Linux Has a Kernel Programming Interface 11
- Linux Can Support Many Users 12
- Linux Can Run Many Tasks 12
- Linux Provides a Secure Hierarchical Filesystem 12
- The Shell: Command Interpreter and Programming Language 13
- X Window System 15
- GUIs: Graphical User Interfaces 16
- A Large Collection of Useful Utilities 18
- Interprocess Communication 18
- (Inter)Networking Utilities 18
- System Administration 19
- Software Development 19

Choosing an Operating System 19

Chapter Summary 21

Exercises 21

PART I Installing Ubuntu Linux 23

Chapter 2: Installation Overview 25

Conventions Used in This Book 26
    LPI and CompTIA Certification Icons 28
More Information 29
Planning the Installation 30
    Considerations 30
    Requirements 30
    Processor Architecture 32
    Interfaces: Installer and Installed System 33
    Gathering Information About the System 34
Ubuntu Releases, Editions, and Derivatives 35
    Ubuntu Releases 35
    Ubuntu Standard Editions: Desktop and Server Images 35
    Ubuntu Derivatives 37
Setting Up the Hard Disk 38
    Primary, Extended, and Logical Partitions 39
    The Linux Directory Hierarchy 40
    Mount Points 40
    Partitioning a Disk 41
    RAID 45
    LVM: Logical Volume Manager 46
Downloading an Image File and Burning/Writing the Installation Medium 47
    The Easy Way to Download an Installation Image File 47
    Other Ways to Download an Installation Image File 48
    Verifying an Installation Image File 51
    Burning a DVD 52
    Writing to a USB Flash Drive 52
Chapter Summary 53
Exercises 54
Advanced Exercises 54

Chapter 3: Step-by-Step Installation 55

Booting Ubuntu and Running a Live Session 56
    Automatic Boot Sequence 56
    Displaying the Boot Menu 57
    Running a Live Session 59
Basic Installation 59
    Installing from a Live Session 60
    Installing from the Desktop Boot Menu 61
    The ubiquity Graphical Installer 61

- The ubiquity Advanced Partitioning Screen 67
- Initializing Databases and Updating the System 70

Advanced Installation 71
- The Boot Menus 72
- Modifying Boot Parameters (Options) 75
- Rescue Mode: Rescuing a Broken System 77
- debian-installer: The Ubuntu Textual Installer 78
- gnome-disks: The GNOME Disk Utility 88
- Setting Up a Dual-Boot System 91

Chapter Summary 93

Exercises 94

Advanced Exercises 94

PART II USING UBUNTU LINUX 95

CHAPTER 4: INTRODUCTION TO UBUNTU 97

Curbing Your Power: **root** Privileges/sudo 98

Logging In on the System 99
- The Unity Desktop 100
- Installing the GNOME Flashback Desktop 103

Working with the Unity Desktop 104
- Terminology 105
- The Dash and the Run a Command Window 106
- Context Menus 106
- Windows 106
- Cutting and Pasting Objects Using the Clipboard 107
- Logging Out 108

Using the Nautilus File Manager 108
- The Nautilus File Browser Window 109
- The Sidebar 109
- Opening Files 110
- Selecting Objects 110
- The Object Properties Window 111

The System Settings Window 113
- Desktop Appearance 115
- Displays 116
- Mouse & Touchpad 116
- Time & Date 117
- User Accounts: Changing Your Account Type and Password (GUI) 118

Getting Help 118
- The Ubuntu Desktop Guide 118
- Using the Internet to Get Help 119

Installing, Removing, and Updating Software Packages 121
- Software & Updates Window 122
- Updating Software 123
- Adding and Removing Software 124
- Installing Other Desktop Environments 124

Working from the Command Line 125
- Running Commands from the Command Line 126
- The Shell 128
- Running Basic Command-Line Utilities 132
- Writing and Executing a Basic Shell Script 134
- Getting Help from the Command Line 135

More About Logging In and Passwords 142
- What to Do If You Cannot Log In 142
- Password Security 143
- passwd: Changing Your Password (CLI) 144

Chapter Summary 145

Exercises 146

Advanced Exercises 147

Chapter 5: The Shell 149

Special Characters 150

Ordinary Files and Directory Files 151
- The Working Directory 151
- Your Home Directory 151

The Command Line 152
- A Simple Command 152
- Syntax 152
- Simple Commands 155
- Processing the Command Line 156
- Executing a Command 158
- Editing the Command Line 159

Standard Input and Standard Output 159
- The Screen as a File 160
- The Keyboard and Screen as Standard Input and Standard Output 160
- Redirection 161
- Pipelines 166
- Lists 170

Running a Command in the Background 171
- Moving a Job from the Foreground to the Background 172
- kill: Aborting a Background Job 172

Filename Generation/Pathname Expansion 173
- The ? Special Character 173

The * Special Character 174
The [ ] Special Characters 176
Builtins 178
Chapter Summary 178
Utilities and Builtins Introduced in This Chapter 179
Exercises 179
Advanced Exercises 181

CHAPTER 6: THE LINUX FILESYSTEM 183

The Hierarchical Filesystem 184
Ordinary Files and Directory Files 185
Filenames 186
Pathnames 189
Absolute Pathnames 189
Relative Pathnames 190
Working with Directories 191
mkdir: Creates a Directory 192
cd: Changes to Another Working Directory 193
rmdir: Deletes a Directory 194
Using Pathnames 195
mv, cp: Move or Copy Files 195
mv: Moves a Directory 196
Important Standard Directories and Files 197
Access Permissions 199
ls –l: Displays Permissions 199
chmod: Changes File Access Permissions 201
chown: Changes File Ownership 203
chgrp: Changes File Group Association 203
Setuid and Setgid Permissions 204
Directory Access Permissions 205
ACLs: Access Control Lists 206
Enabling ACLs 207
Working with Access Rules 207
Setting Default Rules for a Directory 210
Links 211
Hard Links 212
Symbolic Links 214
rm: Removes a Link 216
Chapter Summary 217
Exercises 219
Advanced Exercises 220

Chapter 7: The Linux Utilities 223

Basic Utilities 224
- cat: Joins and Displays Files 224
- date: Displays the System Time and Date 226
- echo: Displays Arguments 227
- hostname: Displays the System Name 227
- less Is more: Display a Text File One Screen at a Time 228
- ls: Displays Information About Files 229
- rm: Removes a File (Deletes a Link) 231

Working with Files 232
- cp: Copies Files 232
- cut: Selects Characters or Fields from Input Lines 233
- diff: Displays the Differences Between Two Text Files 235
- file: Displays the Classification of a File 237
- find: Finds Files Based on Criteria 237
- grep: Searches for a Pattern in Files 240
- head: Displays the Beginning of a File 243
- lpr: Sends Files to Printers 243
- mv: Renames or Moves a File 245
- sort: Sorts and/or Merges Files 247
- tail: Displays the Last Part of a File 249
- touch: Changes File Modification and Access Times 251
- wc: Displays the Number of Lines, Words, and Bytes in Files 252

Compressing and Archiving Files 253
- xz, bzip2, and gzip: Compress and Decompress Files 253
- tar: Stores or Extracts Files to/from an Archive File 257

Displaying User and System Information 260
- free: Displays Memory Usage Information 261
- uptime: Displays System Load and Duration Information 261
- w: Lists Users on the System 262
- who: Lists Users on the System 262

Miscellaneous Utilities 263
- which and whereis: Locate a Utility 263
- locate: Searches for a File 264
- script: Records a Shell Session 265
- tr: Replaces Specified Characters 266
- unix2dos: Converts Linux Files to Windows and Macintosh Format 268
- xargs: Converts Standard Input to Command Lines 268

Editing Files 270
- Tutorial: Using vim to Create and Edit a File 270
- Tutorial: Using nano to Create and Edit a File 277

Chapter Summary 280

Exercises 282

Advanced Exercises 283

Chapter 8: Networking and the Internet 285

Introduction to Networking 286
Types of Networks and How They Work 288
  Broadcast Networks 288
  Point-to-Point Networks 289
  Switched Networks 289
  LAN: Local Area Network 290
  WAN: Wide Area Network 293
  Internetworking Through Gateways and Routers 293
  Network Protocols 296
  IPv4 298
  IPv6 299
  Host Address 302
  CIDR: Classless Inter-Domain Routing 306
  Hostnames 306
Communicate over a Network 307
  Mailing List Servers 307
Network Utilities 308
  Trusted Hosts 308
  OpenSSH Tools 309
  telnet: Logs In on a Remote System 309
  ftp: Transfers Files over a Network 311
  ping: Tests a Network Connection 311
  traceroute: Traces a Route over the Internet 312
  host and dig: Query Internet Nameservers 313
  whois: Looks Up Information About an Internet Site 314
Distributed Computing 315
  The Client/Server Model 315
  DNS: Domain Name Service 316
  Ports 318
  NIS: Network Information Service 319
  NFS: Network Filesystem 319
  Network Services 319
  Common Daemons 320
  Proxy Servers 322
  RPC Network Services 323
WWW: World Wide Web 325
  Browsers 326
  Search Engines 326
  URL: Uniform Resource Locator 326
Chapter Summary 327
Exercises 328
Advanced Exercises 328

PART III SYSTEM ADMINISTRATION 331

CHAPTER 9: THE BOURNE AGAIN SHELL (bash) 333

Background 334
Startup Files 335
  Login Shells 336
  Interactive Nonlogin Shells 336
  Noninteractive Shells 337
  Setting Up Startup Files 337
  . (Dot) or source: Runs a Startup File in the Current Shell 338
Commands That Are Symbols 339
Redirecting Standard Error 339
Writing and Executing a Shell Script 342
  chmod: Makes a File Executable 343
  #! Specifies a Shell 344
  # Begins a Comment 346
  Executing a Shell Script 346
Control Operators: Separate and Group Commands 347
  ; and NEWLINE Separate Commands 347
  | and & Separate Commands and Do Something Else 348
  && and || Boolean Control Operators 349
  ( ) Groups Commands 350
  \ Continues a Command 351
Job Control 352
  jobs: Lists Jobs 352
  fg: Brings a Job to the Foreground 353
  Suspending a Job 354
  bg: Sends a Job to the Background 354
Manipulating the Directory Stack 355
  dirs: Displays the Stack 355
  pushd: Pushes a Directory on the Stack 356
  popd: Pops a Directory off the Stack 357
Parameters and Variables 358
  User-Created Variables 359
  Variable Attributes 362
  Keyword Variables 364
Special Characters 372
Locale 374
  LC_: Locale Variables 374
  locale: Displays Locale Information 375
Time 377
Processes 379
  Process Structure 379

Process Identification 380
Executing a Command 381
History 382
Variables That Control History 383
Reexecuting and Editing Commands 384
The Readline Library 392
Aliases 398
Single Versus Double Quotation Marks in Aliases 399
Examples of Aliases 400
Functions 402
Controlling bash: Features and Options 404
bash Command-Line Options 405
Shell Features 405
Processing the Command Line 409
History Expansion 410
Alias Substitution 410
Parsing and Scanning the Command Line 410
Command-Line Expansion 410
Chapter Summary 420
Exercises 421
Advanced Exercises 423

CHAPTER 10: SYSTEM ADMINISTRATION: CORE CONCEPTS 425

The Upstart Event-Based **init** Daemon 427
Software Package 428
Terminology 428
Jobs 430
SysVinit (**rc**) Scripts: Start and Stop System Services 435
System Operation 437
Runlevels 438
Booting the System 438
Going to Graphical Multiuser Mode 439
Logging In 440
Logging Out 441
Bringing the System Down 441
Crash 443
Using Loadable Kernel Modules 444
GRUB: The Linux Boot Loader 444
Terminology 445
Configuring GRUB 445
grub-mkconfig: Generates the **grub.cfg** File 449
grub-install: Installs the MBR and GRUB Files 450

Recovery (Single-User) Mode 450
    Booting the System to Recovery (Single-User) Mode 451
Textual System Administration Utilities 454
Setting Up a Server 460
    Standard Rules in Configuration Files 461
    rpcinfo: Displays Information About rpcbind 463
DHCP: Configures Network Interfaces 464
    How DHCP Works 465
    DHCP Client 465
    DHCP Server 466
nsswitch.conf: Which Service to Look at First 468
    Information 469
    Methods 469
    Search Order 469
    Action Items 470
    compat Method: **±** in **passwd**, **group**, and **shadow** Files 471
X Window System 471
    Starting X from a Character-Based Display 472
    Remote Computing and Local Displays 472
    Stopping the X Server 475
    Remapping Mouse Buttons (CLI) 475
Getting Help 476
Chapter Summary 477
Exercises 478
Advanced Exercises 478

Chapter 11: Files, Directories, and Filesystems 479

Important Files and Directories 480
File Types 493
    Ordinary Files, Directories, Links, and Inodes 493
    Device Special Files 494
Filesystems 497
    mount: Mounts a Filesystem 499
    umount: Unmounts a Filesystem 501
    du: Displays Disk Usage Information 501
    fstab: Keeps Track of Filesystems 502
    fsck: Checks Filesystem Integrity 503
    tune2fs: Changes Filesystem Parameters 504
The XFS Filesystem 506
Chapter Summary 507
Exercises 508
Advanced Exercises 508

Chapter 12: Finding, Downloading, and Installing Software 509

Introduction 510
JumpStart: Installing and Removing Software Packages Using apt-get 512
Finding the Package That Holds an Application or File You Need 514
APT: Keeps the System Up to Date 515
- Repositories 515
- **sources.list**: Specifies Repositories for APT to Search 516
- The APT Local Package Indexes and the APT Cache 518
- The **apt** cron Script and APT Configuration Files 518
- apt-get: Works with Packages and the Local Package Index 519
- apt-cache: Displays Package Information 522
- apt-get **source**: Downloads Source Files 523

dpkg: The Debian Package Management System 524
- **deb** Files 524
- dpkg: The Foundation of the Debian Package Management System 526

BitTorrent 531
- Prerequisites 531
- transmission-cli: Downloading a BitTorrent File 532

Installing Non-**dpkg** Software 533
- The **/opt** and **/usr/local** Directories 534
- GNU Configure and Build System 534

Keeping Software Up to Date 535
- Bugs 535

curl: Downloads Files Noninteractively 536
Chapter Summary 536
Exercises 537
Advanced Exercises 537

Chapter 13: Printing with CUPS 539

Introduction 540
- Prerequisites 541
- More Information 541
- Notes 542

The System Configures a Local Printer Automatically 542
JumpStart I: Configuring a Printer Using system-config-printer 542
- Configuration Tabs 543

JumpStart II: Setting Up a Local or Remote Printer 544
Working with the CUPS Web Interface 548

Configuring Printers 549
Modifying a Printer 549
Using the CUPS Web Interface 550
CUPS on the Command Line 551
Sharing CUPS Printers 555
Traditional UNIX Printing 557
Printing from Windows 558
Printing Using CUPS 558
Printing Using Samba 559
Printing to Windows 560
Chapter Summary 560
Exercises 561
Advanced Exercises 561

Chapter 14: Administration Tasks 563

Configuring User and Group Accounts 564
unity-control-center: Manages User Accounts 564
Managing User Accounts from the Command Line 566
Backing Up Files 568
Choosing a Backup Medium 569
Backup Utilities 569
Performing a Simple Backup 572
Scheduling Tasks 573
cron and anacron: Schedule Routine Tasks 573
at: Runs Occasional Tasks 576
System Reports 576
vmstat: Reports Virtual Memory Statistics 576
top: Lists Processes Using the Most Resources 577
Maintaining the System 578
timedatectl: Reports on and Sets the System Clock 579
parted: Reports on and Partitions a Hard Disk 579
logrotate: Manages Log Files 582
rsyslogd: Logs System Messages 585
Solving Problems 587
Chapter Summary 593
Exercises 594
Advanced Exercises 594

Chapter 15: System Security 595

Running Commands with **root** Privileges 596
Administrator 596
The Special Powers of a User Working with **root** Privileges 596

Gaining **root** Privileges 597
Real UID Versus Effective UID 599
Using su to Gain **root** Privileges 600
Using sudo to Gain **root** Privileges 602
Unlocking the **root** Account (Assigning a Password to **root**) 613
Avoiding a Trojan Horse 613
Passwords 615
Securing a Server 616
TCP Wrappers 616
Setting Up a chroot Jail 617
PAM 621
Cryptography 626
Features 626
Terminology 627
Encrypting a Message 627
Cryptographic Hash Functions 632
Signing a Message Using a Hash Value 636
SSL Certificates 637
GPG (GNU Privacy Guard) 641
Tutorial: Using GPG to Secure a File 641
gpg-agent: Holds Your Private Keys 648
Make Your Communication More Secure 648
Encrypting and Decrypting a File 649
Signing and Encrypting a File 650
Signing a Key on Your Keyring 651
Using a Keyserver 652
Security Resources 656
Chapter Summary 659
Exercises 660
Advanced Exercises 660

Chapter 16: Configuring and Monitoring a LAN 661

More Information 662
Setting Up the Hardware 662
Connecting the Computers 662
Routers 663
NIC: Network Interface Card 664
Tools 664
Configuring the Systems 666
NetworkManager: Configures Network Connections 667
The NetworkManager Applet Menu 668
Setting Up Servers 672

Introduction to Cacti 674
  Configuring SNMP 675
  Setting Up LAMP 675
  Prerequisites 675
  Configuring MySQL 676
  Setting Up Cacti 677
  Configuring Cacti 678
  Basic Cacti Administration 680
  Setting Up a Data Source 681
Chapter Summary 683
Exercises 684
Advanced Exercises 685

Chapter 17: Setting Up Virtual Machines Locally and in the Cloud 687

VMs (Virtual Machines) 688
  Implementations 690
gnome-boxes 690
QEMU/KVM 691
  Prerequisites 692
  virt-manager: Installing Ubuntu on QEMU/KVM 693
  virsh: Work with VMs on the Command Line 697
VMware Player: Installing Ubuntu on VMware 698
  Installing VMware Player on a Linux System 699
  Installing VMware Player on a Windows System 700
  Installing Ubuntu on VMware Player 700
Cloud Computing 703
  AWS: Setting Up a Virtual System in the Cloud 704
Chapter Summary 708
Exercises 709
Advanced Exercises 709

PART IV Using Clients and Setting Up Servers 711

Chapter 18: The OpenSSH Secure Communication Utilities 713

Introduction to OpenSSH 714
  Files 714
  More Information 716
Running the ssh, scp, and sftp OpenSSH Clients 716
  Prerequisites 716

JumpStart I: Using ssh and scp to Connect to an OpenSSH Server 716
Configuring OpenSSH Clients 717
ssh: Logs in or Executes Commands on a Remote System 720
scp: Copies Files to and from a Remote System 723
sftp: A Secure FTP Client 725
~/.ssh/config and **/etc/ssh/ssh_config** Configuration Files 726
Setting Up an OpenSSH Server (**sshd**) 727
Prerequisites 727
Note 728
JumpStart II: Starting an OpenSSH Server 728
Authorized Keys: Automatic Login 728
Randomart Image 730
ssh-agent: Holds Your Private Keys 731
Command-Line Options 732
/etc/ssh/sshd_config Configuration File 732
Troubleshooting 735
Tunneling/Port Forwarding 735
Forwarding X11 736
Port Forwarding 737
Chapter Summary 738
Exercises 739
Advanced Exercises 739

Chapter 19: The rsync Secure Copy Utility 741

Syntax 742
Arguments 742
Options 742
Notes 744
More Information 745
Examples 745
Using a Trailing Slash (/) on ***source-file*** 745
Removing Files 746
Copying Files to and from a Remote System 748
Mirroring a Directory 748
Making Backups 749
Restoring a File 752
Chapter Summary 752
Exercises 752

Chapter 20: FTP: Transferring Files Across a Network 753

Introduction to FTP 754
Security 754
FTP Connections 755
FTP Clients 755
More Information 755
Notes 756

Running the ftp and sftp FTP Clients 756
- Prerequisites 756
- JumpStart I: Downloading Files Using ftp 756
- Anonymous FTP 759
- Automatic Login 760
- Binary Versus ASCII Transfer Mode 760
- ftp Specifics 761

Setting Up an FTP Server (**vsftpd**) 764
- Prerequisites 764
- Notes 765
- JumpStart II: Starting a **vsftpd** FTP Server 765
- Troubleshooting 765
- Configuring a **vsftpd** Server 766

Chapter Summary 777

Exercises 778

Advanced Exercises 778

Chapter 21: postfix: Setting Up Mail Servers, Clients, and More 779

Overview 780
- Mailboxes: **mbox** Versus **maildir** Format 780
- Protocols: IMAP and POP3 781

Introduction to **postfix** 781
- Outbound Email 782
- Inbound Email 782
- The **postfix** to **sendmail** Compatibility Interface 782
- Alternatives to postfix 783
- More Information 783

Setting Up a postfix Mail Server 784
- Prerequisites 784
- Notes 784
- Testing **postfix** 785
- **postfix** Log Files 786

JumpStart: Configuring **postfix** to Use Gmail as a Smarthost 787

Configuring **postfix** 789
- The **/etc/mailname** File 789
- The **/etc/postfix/main.cf** File 789
- **postfix** Lookup Tables 793
- The **/etc/postfix/master.cf** File 794
- Aliases and Forwarding 794
- dpkg-reconfigure: Reconfigures **postfix** 796

SpamAssassin 797
- How SpamAssassin Works 797
- Prerequisites 797
- Testing SpamAssassin 798
- Configuring SpamAssassin 799

Additional Email Tools 801
- Webmail 801
- Mailing Lists 804

dovecot: Setting Up an IMAP or POP3 Mail Server 807
- Prerequisites 807
- Notes 808
- Testing **dovecot** 808

Chapter Summary 810

Exercises 811

Advanced Exercises 812

Chapter 22: NIS and LDAP 813

Introduction to NIS 814
- How NIS Works 814
- More Information 816

Running an NIS Client 817
- Prerequisites 817
- Notes 818
- Configuring an NIS Client 818
- Troubleshooting an NIS Client 820
- yppasswd: Changes NIS Passwords 821

Setting Up an NIS Server 822
- Prerequisites 823
- Notes 823
- Configuring an NIS Server 824
- Troubleshooting an NIS Server 829
- **yppasswdd**: The NIS Password Update Daemon 830

Introduction to LDAP 830
- More Information 833

Setting Up an LDAP Server 833
- Prerequisites 833
- Notes 834
- Test the Server 834
- Modifying and Adding Directory Entries 834
- Using Thunderbird with LDAP 838

Chapter Summary 839

Exercises 840

Advanced Exercises 840

Chapter 23: NFS: Sharing Directory Hierarchies 843

Introduction to NFS 845
- More Information 847

Running an NFS Client 847
- Prerequisites 847
- JumpStart I: Mounting a Remote Directory Hierarchy 848
- mount: Mounts a Directory Hierarchy 849
- Improving Performance 852
- **/etc/fstab**: Mounts Directory Hierarchies Automatically 853

Setting Up an NFS Server 853
- Prerequisites 853
- Notes 854
- JumpStart II: Configuring an NFS Server Using shares-admin 855
- Manually Exporting a Directory Hierarchy 857
- Where the System Keeps NFS Mount Information 860
- exportfs: Maintains the List of Exported Directory Hierarchies 861
- Troubleshooting 862

automount: Mounts Directory Hierarchies on Demand 863

Chapter Summary 866

Exercises 867

Advanced Exercises 867

Chapter 24: Samba: Linux and Windows File and Printer Sharing 869

Introduction to Samba 870
- More Information 871
- Notes 871
- Samba Users, User Maps, and Passwords 871
- smbpasswd and pdbedit: Work with Samba Users and Passwords 873

Running Samba Clients 874
- Prerequisites 874
- Working with Shares from Linux 874
- Working with Shares from Windows 877

Setting Up a Samba Server 878
- Prerequisites 878
- JumpStart: Configuring a Samba Server Using shares-admin 879
- **smb.conf**: Manually Configuring a Samba Server 880

Troubleshooting 887

Chapter Summary 889

Exercises 890

Advanced Exercises 890

Chapter 25: DNS/BIND: Tracking Domain Names and Addresses 891
Introduction to DNS 892
 Nodes, Domains, and Subdomains 893
 Zones 895
 Queries 896
 Servers 896
 Resource Records 897
 DNS Queries and Responses 901
 Reverse Name Resolution 902
 How DNS Works 903
 More Information 904
Setting Up a DNS Server 904
 Prerequisites 904
 Notes 905
 JumpStart: Setting Up a DNS Cache 906
Configuring a DNS Server 907
 named.conf: The **named** Configuration File 907
 Zone Files 910
 Setting Up a DNS Cache 911
 DNS Glue Records 914
 TSIGs: Transaction Signatures 915
 Running BIND in a chroot Jail 917
 Troubleshooting 919
Chapter Summary 920
Exercises 921
Advanced Exercises 921

Chapter 26: ufw, gufw, and iptables: Setting Up a Firewall 923
ufw: The Uncomplicated Firewall 924
gufw: The Graphical Interface to ufw 927
 The Firewall Window 927
 Adding Rules 928
Introduction to iptables 932
 More Information 935
 Prerequisites 935
 Notes 935
 Anatomy of an iptables Command 936
Building a Set of Rules Using iptables 937
 Commands 938
 Packet Match Criteria 939
 Display Criteria 940
 Match Extensions 940
 Targets 942

Copying Rules to and from the Kernel 944
Sharing an Internet Connection Using NAT 945
  Connecting Several Clients to a Single Internet Connection 946
  Connecting Several Servers to a Single Internet Connection 948
Chapter Summary 948
Exercises 949
Advanced Exercises 949

Chapter 27: Apache (apache2): Setting Up a Web Server 951

Introduction 952
  More Information 952
  Notes 953
Running an Apache Web Server 954
  Prerequisites 954
  JumpStart: Getting Apache Up and Running 955
  Configuring Apache 957
  Filesystem Layout 959
Configuration Directives 961
  Directives You Might Want to Modify as You Get Started 962
  Contexts and Containers 966
  Advanced Configuration Directives 971
Advanced Configuration 984
  Redirects 984
  Content Negotiation 985
  Server-Generated Directory Listings (Indexing) 986
  Virtual Hosts 986
Troubleshooting 990
Modules 991
  mod_cgi and CGI Scripts 992
  mod_ssl 992
  Authentication Modules and **.htaccess** Files 994
  Scripting Modules 995
  Multiprocessing Modules (MPMs) 996
webalizer: Analyzes Web Traffic 997
Error Codes 997
Chapter Summary 998
Exercises 998
Advanced Exercises 999

PART V Programming Tools 1001

Chapter 28: Programming the Bourne Again Shell (bash) 1003

Control Structures 1004
- **if...then** 1005
- **if...then...else** 1009
- **if...then...elif** 1011
- **for...in** 1017
- **for** 1019
- **while** 1021
- **until** 1025
- **break** and **continue** 1027
- **case** 1028
- **select** 1034
- Here Document 1036

File Descriptors 1038
- Opening a File Descriptor 1039
- Duplicating a File Descriptor 1039
- File Descriptor Examples 1039
- Determining Whether a File Descriptor Is Associated with the Terminal 1042

Parameters 1044
- Positional Parameters 1044
- Special Parameters 1049

Variables 1053
- Shell Variables 1053
- Environment, Environment Variables, and Inheritance 1054
- Expanding Null and Unset Variables 1058
- Array Variables 1060
- Variables in Functions 1061

Builtin Commands 1062
- type: Displays Information About a Command 1063
- read: Accepts User Input 1063
- exec: Executes a Command or Redirects File Descriptors 1067
- trap: Catches a Signal 1069
- kill: Aborts a Process 1072
- eval: Scans, Evaluates, and Executes a Command Line 1073
- getopts: Parses Options 1074
- A Partial List of Builtins 1077

Expressions 1078
- Arithmetic Evaluation 1078

Logical Evaluation (Conditional Expressions) 1079
String Pattern Matching 1080
Arithmetic Operators 1081
Implicit Command-Line Continuation 1085
Shell Programs 1086
A Recursive Shell Script 1087
The **quiz** Shell Script 1090
Chapter Summary 1096
Exercises 1098
Advanced Exercises 1100

Chapter 29: The Python Programming Language 1103

Introduction 1104
Invoking Python 1104
More Information 1106
Writing to Standard Output and Reading from Standard Input 1107
Functions and Methods 1107
Scalar Variables, Lists, and Dictionaries 1108
Scalar Variables 1108
Lists 1109
Dictionaries 1113
Control Structures 1114
if 1115
if...else 1115
if...elif...else 1116
while 1117
for 1117
Reading from and Writing to Files 1119
File Input and Output 1119
Exception Handling 1120
Pickle 1122
Regular Expressions 1123
Defining a Function 1124
Using Libraries 1125
Standard Library 1125
Nonstandard Libraries 1125
SciPy and NumPy Libraries 1126
Namespace 1126
Importing a Module 1127
Example of Importing a Function 1128
Lambda Functions 1129

List Comprehensions 1130
Chapter Summary 1131
Exercises 1132
Advanced Exercises 1132

Chapter 30: The MariaDB SQL Database Management System 1135

History 1136
Notes 1136
- Terminology 1137
- Syntax and Conventions 1138
- More Information 1139

Installing a MariaDB Server 1140
- Client Options 1140

Setting Up MariaDB 1141
- Assigning a Password to the MariaDB User Named **root** 1141
- Removing Anonymous Users 1141
- Running the Secure Installation Script 1142
- **~/.my.cnf**: Configures a MariaDB Client 1142
- **~/.mysql_history**: Stores Your MariaDB History 1142
- Creating a Database 1143
- Adding a User 1144

Examples 1145
- Logging In 1145
- Creating a Table 1145
- Adding Data 1147
- Retrieving Data 1148
- Backing Up a Database 1150
- Modifying Data 1150
- Creating a Second Table 1151
- Joins 1152

Chapter Summary 1157
Exercises 1157
Advanced Exercises 1157

PART VI Appendixes 1159

Appendix A: Regular Expressions 1161

Characters 1162
Delimiters 1162

Simple Strings 1162
Special Characters 1162
  Periods 1163
  Brackets 1163
  Asterisks 1164
  Carets and Dollar Signs 1164
  Quoting Special Characters 1165
Rules 1165
  Longest Match Possible 1165
  Empty Regular Expressions 1166
Bracketing Expressions 1166
The Replacement String 1166
  Ampersand 1167
  Quoted Digit 1167
Extended Regular Expressions 1167
Appendix Summary 1169

Appendix B: Help 1171

Solving a Problem 1172
Finding Linux-Related Information 1173
  Desktop Applications 1173
  Programming Languages 1174
  Linux Newsgroups 1174
  Mailing Lists 1175
Specifying a Terminal 1175

Appendix C: Keeping the System Up to Date Using yum 1177

Installing and Removing Software Packages Using yum 1178
Working with yum 1179
  Finding the Package That Holds a File You Need 1179
  Updating Packages 1180
  yum Commands 1181
  yum.conf: Configures yum 1182
  yum Repositories 1182

Appendix D: LPI and Comptia Certification 1183

More Information 1184
Linux Essentials 1184
  Topic 1: The Linux Community and a Career in Open Source 1184

Topic 2: Finding Your Way on a Linux System 1187
Topic 3: The Power of the Command Line 1190
Topic 4: The Linux Operating System 1192
Topic 5: Security and File Permissions 1195
Certification Exam 1 Objectives: LX0-101 1198
101 System Architecture 1198
102 Linux Installation and Package Management 1200
103 GNU and Unix Commands 1203
104 Devices, Linux Filesystems, Filesystem Hierarchy Standard 1210
Certification Exam 2 Objectives: LX0-102 1214
105 Shells, Scripting and Data Management 1214
106 User Interfaces and Desktops 1217
107 Administrative Tasks 1218
108 Essential System Services 1221
109 Networking Fundamentals 1223
110 Security 1226

Glossary 1231

JumpStart Index 1285

File Tree Index 1287

Utility Index 1291

Main Index 1297

This page intentionally left blank

Preface

The book Whether you are an end user, a system administrator, or a little of both, this book explains with step-by-step examples how to get the most out of an Ubuntu system. In 30 chapters, this book takes you from installing an Ubuntu system, through understanding its inner workings, to setting up secure servers that run on the system.

The audience This book is designed for a wide range of readers. It does not require you to have programming experience, although having some experience using a general-purpose computer, such as a Windows, Macintosh, UNIX, or another Linux system is certainly helpful. This book is appropriate for:

- **Students** who are taking a class in which they use Linux
- **Home users** who want to set up and/or run Linux
- **Professionals** who use Linux at work
- **System administrators** who need an understanding of Linux and the tools that are available to them, including the bash and Python scripting languages
- **Computer science students** who are studying the Linux operating system
- **Technical executives** who want to get a grounding in Linux

Benefits *A Practical Guide to Ubuntu Linux®, Fourth Edition,* gives you a broad understanding of many facets of Linux, from installing Ubuntu through using and customizing it. No matter what your background, this book provides the knowledge you need to get on with your work. You will come away from this book understanding how to use Linux, and this book will remain a valuable reference for years to come.

Features in this edition

This edition covers many topics to help you get your work done using Ubuntu.

- Full coverage of LPI's Linux Essentials certification learning goals and extensive coverage of CompTIA's Linux+ exam objectives (Appendix D; page 1183)
- Updated chapters reflecting new features in Ubuntu 14.04 (Trusty Tahr)—the LTS (Long Term Support) release Canonical will support into 2019
- A new chapter that covers setting up VMs (virtual machines) and working in the cloud (Chapter 17; page 687)
- A new chapter on the Python programming language (Chapter 29; page 1103)
- A new chapter on system security (Chapter 15; page 595)
- A new chapter covering 32 Linux utilities (Chapter 7; page 223)
- A new chapter on the MariaDB/MySQL relational database (Chapter 30; page 1135)
- Updated coverage of the ufw and gufw firewall utilities (Chapter 26; page 924)
- Tutorials on the vim and nano editors (Chapter 7; pages 270 and 277)
- Nine chapters on system administration (Part III; page 331)
- A chapter on writing programs using bash (Chapter 28; page 1003)
- Coverage of the XFS filesystem (Chapter 11; page 506)
- Coverage of LDAP (Chapter 22; page 830)
- A section on the Cacti network monitoring tool (Chapter 16; page 674)
- Coverage of IPv6 (Chapter 8; page 299)
- Four indexes, making it easier to quickly find what you are looking for. These indexes locate tables (page numbers followed by the letter **t**), provide definitions (italic page numbers), and differentiate between light and comprehensive coverage (light and standard fonts).
    - The JumpStart index (page 1285) lists all JumpStart sections in this book. These sections help you set up servers and clients quickly.
    - The File Tree index (page 1287) lists, in hierarchical fashion, most files mentioned in this book. These files are also listed in the Main index.
    - The Utility index (page 1291) supplies the location of all utilities mentioned in this book. A page number in a light font indicates a brief mention of the utility, whereas the regular font indicates more substantial coverage. The Utility index also appears on the inside of the front and back covers of the print book.
    - The revised Main index (page 1297) is designed for ease of use.

Overlap If you have read *A Practical Guide to Linux® Commands, Editors, and Shell Programming, Third Edition,* you will notice some overlap between that book and the one you are reading now. The first chapter, the chapters on the utilities, the filesystem, and rsync, the appendix on regular expressions, and the Glossary are very similar in the two books, as are the three chapters on the Bourne Again Shell (bash) and the chapters on Python and MariaDB. Chapters that appear in this book but do not appear in *A Practical Guide to Linux® Commands, Editors, and Shell Programming, Third Edition,* include Chapters 2 and 3 (installation), Chapter 4 (Ubuntu and the GUI), Chapter 8 (networking), and all of the chapters in Part III (system administration) and Part IV (servers).

Differences While this book explains how to use Linux from a graphical interface and from the command line (a textual interface), *A Practical Guide to Linux® Commands, Editors, and Shell Programming, Third Edition,* works exclusively with the command line and covers Mac OS X in addition to Linux. It includes full chapters on the vim and emacs editors, as well as chapters on the gawk pattern processing language and the sed stream editor. In addition, it has a command reference section that provides extensive examples of the use of 98 of the most important Linux and Mac OS X utilities. You can use these utilities to solve problems without resorting to programming in C.

This Book Includes an Ubuntu 14.04 (Trusty Tahr) DVD

The print book includes a DVD that holds a Desktop Image (installation image) of Ubuntu 14.04 (Trusty Tahr). You can use this DVD to install an Ubuntu 14.04 desktop system. Chapter 2 helps you get ready to install Ubuntu. Chapter 3 provides step-by-step instructions for installing Ubuntu from this DVD. This book guides you through learning about, using, and administrating an Ubuntu system.

Live system In addition to installing Ubuntu from the DVD, you can use the DVD to run a live Ubuntu session that displays the Unity desktop without making any changes to your computer: Boot from the DVD, run an Ubuntu live session, and log off. Your system remains untouched: When you reboot, it is exactly as it was before you ran the Ubuntu live session. For more information refer to “Booting Ubuntu and Running a Live Session” on page 56.

DVD features The Desktop Image DVD includes many of the software packages supported by Ubuntu. You can use it to perform a graphical installation of a graphical Ubuntu system. If you do not have an Internet connection, you can use the DVD as a software repository: After you have installed Ubuntu, you can install supported software packages from the DVD.

For Readers of Digital Editions

If you are reading a digital edition of this book, see “Downloading an Image File and Burning/Writing the Installation Medium” on page 47 for instructions on how to download an installation image and create a DVD or USB flash drive that holds that image.

Features of This Book

This book is designed and organized so you can get the most out of it in the least amount of time. You do not have to read this book straight through in page order. Instead, once you are comfortable using Linux, you can use this book as a reference: Look up a topic of interest in the table of contents or in an index and read about it. Or think of the book as a catalog of Linux topics: Flip through the pages until a topic catches your eye. The book includes many pointers to Web sites where you can obtain additional information: Consider the Internet to be an extension of this book.

A Practical Guide to Ubuntu Linux®, Fourth Edition, is structured with the following features.

- **Optional sections** enable you to read the book at different levels, returning to more difficult material when you are ready to delve into it.
- **Caution boxes** highlight procedures that can easily go wrong, giving you guidance *before* you run into trouble.
- **Tip boxes** highlight ways you can save time by doing something differently or situations when it might be useful or just interesting to have additional information.
- **Security boxes** point out places where you can make a system more secure. Chapter 15 presents a thorough background in system **security concepts** and issues and includes a tutorial on GPG.
- Concepts are illustrated by **practical examples** throughout the book.
- Each chapter starts with a list of **chapter objectives—**a list of important tasks you should be able to perform after reading the chapter.
- **Chapter summaries** review the important points covered in each chapter.
- **Review exercises** are included at the end of each chapter for readers who want to further hone their skills. Answers to even-numbered exercises are posted at www.sobell.com.
- The **Glossary** defines more than 500 commonly encountered terms.
- The chapters covering servers include **JumpStart** sections that get you off to a quick start using clients and setting up servers. Once a server is up and running, you can test and modify its configuration, as is explained in the rest of each of these chapters.
- This book provides resources for **finding software** on the Internet. It also explains how to **download** and **install** software using apt-get, the Ubuntu Software Center window, and BitTorrent. It details controlling automatic updates using the Software & Updates window.
- This book describes in detail many important **GNU tools**, including the Nautilus File Browser, the parted and gnome-disks partition editors, the gzip

compression utility, and many command-line utilities that come from the GNU Project. It also covers the Unity desktop, Ubuntu's graphical shell for the GNOME desktop environment.

- Pointers throughout the text provide help in obtaining **online documentation** from many sources, including the local system, the Ubuntu Web sites, and other locations on the Internet.
- Multiple **comprehensive indexes** help you locate topics quickly and easily.

Key Topics Covered in This Book

This section distills and summarizes the information covered by this book. In addition, "Details" (starting on page l) describes what each chapter covers. Finally, the Table of Contents (starting on page xvii) provides more detail. This book:

Installation

- Describes how to download Ubuntu installation image files from the Internet and write or burn the image file to a USB flash drive, CD, or DVD.
- Helps you plan the layout of the system's hard disk. It includes a discussion of partitions, partition tables, and mount points, and explains how to use ubiquity, the gnome-disks disk utility, or the Ubuntu textual partition editor to examine and partition the hard disk.
- Explains how to set up a dual-boot system so you can install Ubuntu on a Windows system and boot either operating system.
- Discusses booting a live Ubuntu session and installing Ubuntu from that session.
- Describes in detail how to install Ubuntu from an installation image using the ubiquity graphical installer. It also explains how to use the textual installer to install Ubuntu. The graphical installer is fast and easy to use. The textual installer gives you more options and works on systems with less RAM (system memory).
- Covers testing installation media for defects, setting boot command-line parameters (boot options), and creating a RAID array.
- Describes how to set up a VM (virtual machine) and install Ubuntu on the VM.
- Describes how the Logical Volume Manager (LVM2) can set up, grow, and migrate logical volumes, which are similar in function to traditional disk partitions.

Working with Ubuntu

- Introduces the Unity desktop (GUI), and explains how to use desktop tools, including application and context menus, the Settings window, the Nautilus File Browser, and the GNOME terminal emulator; also covers installation of the GNOME 3 and GNOME 2 (Classic or Flashback) desktops.

- Covers the Bourne Again Shell (bash) in three chapters, including an entire chapter on shell programming, which includes many sample shell scripts. These chapters provide clear explanations and extensive examples of how bash works both from the command line in day-to-day work and as a programming language in which to write shell scripts.
- Explains the textual (command-line) interface and introduces more than 32 command-line utilities.
- Presents tutorials on the vim and nano textual editors.
- Covers types of networks, network protocols (including IPv6), and network utilities.
- Explains hostnames, IP addresses, and subnets, and explores how to use host and dig to look up domain names and IP addresses on the Internet.
- Covers distributed computing and the client/server model.
- Explains how to use ACLs (Access Control Lists) to fine-tune user access permissions.

System administration

- Explains how to use the Ubuntu graphical and textual (command-line) tools to configure the display, DNS, NFS, Samba, Apache, a firewall, a network interface, and more. You can also use these tools to add users and manage local and remote printers.
- Explains how you can unlock the **root** account if necessary and describes how to use su to work with **root** privileges (become Superuser), and the advantages and dangers of working with escalated privileges.
- Goes into detail about using sudo to allow specific users to work with **root** privileges and customizing the way sudo works by editing the **sudoers** configuration file.
- Describes how to use the following tools to download and install software to keep a system up to date and to install new software:
    - ◆ The **Software & Updates** window controls which Ubuntu and third-party software repositories Ubuntu downloads software packages from and whether Ubuntu downloads updates automatically. You can also use this window to cause Ubuntu to download and install security updates automatically.
    - ◆ Based on how you set up updates in the Software & Updates window, the **Software Updater** window appears on the desktop to let you know when software updates are available. You can download and install updates from the **Software Updater** window.
    - ◆ The **Ubuntu Software Center** window provides an easy way to select, download, and install a wide range of software packages.
    - ◆ **APT** downloads and installs software packages from the Internet (or the included DVD), keeping a system up to date and resolving dependencies as

it processes the packages. You can use APT from the Synaptic graphical interface or from the apt-get textual interface.

- ◆ **BitTorrent** is a good choice for distributing large amounts of data such as Ubuntu installation images. The more people who use BitTorrent to download a file, the faster it works.

- Covers graphical system administration tools, including the many tools available from the **Unity desktop**.
- Explains system operation, including the boot process, **recovery (single-user) and multiuser modes**, and steps to take if the system crashes.
- Details the workings of the **Upstart init daemon**, which replaces the System V **init** daemon.
- Explains how to set up and use the **Cacti** network monitoring tool to graph system and network information over time, including installing and setting up the **LAMP** (Linux, Apache, MariaDB/MySQL, and PHP) stack.
- Provides instructions on installing, setting up, and using a **MariaDB/MySQL** relational database.
- Discusses setting up and repairing an XFS filesystem.
- Describes files, directories, and filesystems, including types of files and filesystems, **fstab** (the filesystem table), and automatically mounted filesystems, and explains how to fine-tune and check the integrity of filesystems.
- Covers backup utilities, including tar and cpio.
- Describes compression/archive utilities, including xz, gzip, bzip2, compress, and zip.

Security

- Helps you manage basic **system security** issues using ssh (secure shell), **vsftpd** (secure FTP server), Apache (Web server), ufw and iptables (firewalls), and more.
- Discusses **cryptography**, including concepts of authentication, confidentiality (secrecy), data integrity, and nonrepudiation of origin.
- Explains how to **encrypt a message** using symmetric/private key and asymmetric/public key encryption as well as how to use a hybrid encryption system.
- Describes how to use a **cryptographic hash function** to verify the integrity of a downloaded file and how a salt helps protect against dictionary attacks.
- Describes how to use OpenSSL to create an **SSL certificate**.
- Covers using **GPG (GNU Privacy Guard)** to provide identification, secrecy, and message integrity in email and file sharing, and includes a tutorial on its use.
- Describes how to use the textual uncomplicated firewall (ufw) and its graphical interface (gufw) to **protect the system**.

- Provides instructions on using iptables to share an Internet connection over a LAN and to **build advanced firewalls**.
- Describes how to set up a chroot jail to help **protect a server system**.
- Explains how to use TCP wrappers to control who can access a server.

Clients and servers

- Explains how to set up and use the most popular Linux servers, providing a chapter on each: Apache; Samba; OpenSSH; **postfix**; DNS; NFS; FTP; ufw, gufw, and iptables; and NIS/LDAP.
- Describes how to set up a **CUPS printer server**.
- Explains how to set up and use a **MariaDB/MySQL** relational database.
- Describes how to set up and use a **DHCP** server.

Programming

- Provides a chapter on the **Python** programming language and a full chapter covering **shell programming** using bash, including many examples.

Details

Chapter 1 **Chapter 1** presents a brief history of Linux and describes some of the features that make it a cutting-edge operating system.

Part I Part I, "Installing Ubuntu Linux," discusses how to install Ubuntu Linux. **Chapter 2** presents an overview of the process of installing Ubuntu Linux, including hardware requirements, downloading and burning or writing the installation medium, and planning the layout of the hard disk. The "Conventions Used in This Book" section on page 26 details the typefaces and terminology used in this book. **Chapter 3** is a step-by-step guide to installing Ubuntu; it covers installing from an installation image, from a live session, and using the textual installer.

Part II Part II, "Using Ubuntu Linux," familiarizes you with Ubuntu, covering logging in, the GUI, utilities, the filesystem, and the shell. **Chapter 4** introduces desktop features; describes configuring the system using the System Settings window; explains how to use the Nautilus File Browser to manage files, run programs, and connect to FTP and HTTP servers; covers dealing with login problems and using the window manager; and presents some suggestions on where to find documentation, including manuals, tutorials, software notes, and HOWTOs. The introduction to the command line includes information on terminal emulators, virtual consoles, correcting mistakes on the command line, a few basic utilities, and how to write and execute a simple shell script. **Chapter 5** introduces the Bourne Again Shell (bash) and discusses command-line arguments and options, redirecting input to and output from commands, running programs in the background, and using the shell to generate and expand filenames. **Chapter 6** discusses the Linux hierarchical filesystem, covering files, filenames, pathnames, working with directories, access permissions, and hard and symbolic links. **Chapter 7** provides in-depth coverage of 32 useful utilities and presents tutorials on the vim and nano text editors. **Chapter 8** explains networks, network security, and the Internet and discusses types of networks, subnets, protocols, addresses, hostnames, and various network utilities. A section covers the all-important IPv6 protocol. The section on distributed computing describes the client/server model and some of the servers you can use on a network. (Details of setting up and using clients and servers are reserved until Part IV.)

Experienced users may want to skim Part II

tip If you have used a UNIX or Linux system before, you may want to skim or skip some or all of the chapters in Part II. Do *not* skip "Conventions Used in This Book" (page 26), which explains the typographic and layout conventions used in this book. Both "Getting Help" (page 118), which explains how to get help using a GUI, and "Getting Help from the Command Line" (page 135) point out both local and remote sources of Linux and Ubuntu documentation.

Part III Part III, "System Administration," goes into more detail about administrating the system. **Chapter 9** extends the bash coverage from Chapter 5, explaining how to redirect error output, avoid overwriting files, and work with job control, processes, startup files, important shell builtin commands, parameters, shell variables, and aliases. **Chapter 10** discusses core concepts of system operation, including a discussion of the Upstart **init** daemon; the GRUB boot loader; general information about how to set up a server; and DHCP. **Chapter 11** explains the Linux filesystem, going into detail about types of files, including special (device) files; the use of fsck both to verify the integrity of filesystems and to repair them; the use of tune2fs to change filesystem parameters; and the XFS filesystem and related utilities. **Chapter 12** explains how to keep a system up to date by downloading software from the Internet and installing it, including examples that use APT programs such as apt-get and apt-cache to perform these tasks. It also covers the **dpkg** software packaging system and the use of some **dpkg** utilities. Finally, it explains how to use BitTorrent from the command line to download files. **Chapter 13** explains how to set up the CUPS printing system so you can print on both local and remote printers. **Chapter 14** covers additional administration tasks, including setting up user accounts, backing up files, scheduling automated tasks, tracking disk usage, and solving general problems. **Chapter 15** covers system security, including using su and sudo to run commands with **root** privileges; securing servers using TCP wrappers, chroot jails, and PAM; how to use cryptography and hashes to secure and verify data; creating and using an SSL certificate; and securing data in transit using GPG (GNU Privacy Guard). **Chapter 16** explains how to set up a local area network (LAN), including both hardware (including wireless) and software configuration and how to set up Cacti to monitor the network. **Chapter 17** describes VMs (virtual machines), how to set up and work with VMs, and how to work with VMs in the cloud.

Part IV Part IV goes into detail about setting up and running servers and connecting to them using clients. Where appropriate, these chapters include JumpStart sections, which get you off to a quick start in using clients and setting up servers. The chapters in Part IV cover the following clients/servers:

- **OpenSSH—**Set up an OpenSSH server and use ssh, scp, and sftp to communicate securely over the Internet.
- **rsync—**Use rsync to copy files securely from one system to another.

- **FTP**—Set up a **vsftpd** secure FTP server and use any of several FTP clients to exchange files with the server.
- **Email**—Configure **postfix** and use Webmail, POP3, or IMAP to retrieve email; use SpamAssassin to combat spam.
- **NIS and LDAP**—Set up NIS to facilitate system administration of a LAN and LDAP to maintain databases.
- **NFS**—Share filesystems between systems on a network.
- **Samba**—Share filesystems and printers between Windows and Linux systems.
- **DNS/BIND**—Set up a domain nameserver to let other systems on the Internet know the names and IP addresses of local systems they may need to contact.
- **ufw**, **gufw**, and **iptables**—Set up a firewall to protect local systems and share a single Internet connection between systems on a LAN.
- **Apache**—Set up an HTTP server that serves Web pages, which browsers can then display. This chapter includes many suggestions for increasing Apache security.

Part V Part V covers three important programming tools that are used extensively in Ubuntu system administration and general-purpose programming. **Chapter 28** continues where Chapter 9 left off, going into greater depth about shell programming using bash, with the discussion enhanced by extensive examples. **Chapter 29** introduces the flexible and friendly Python programming language, including coverage of lists and dictionaries, using libraries, defining functions, regular expressions, and list comprehensions. **Chapter 30** covers the widely used MariaDB/MySQL RDBMS (relational database management system), including installation, creating a database, adding a user, creating and modifying tables, joins, and adding data to the database.

Part VI Part VI includes appendixes on regular expressions, helpful Web sites, updating software using yum, and a map that indexes LPI's Linux Essentials certification learning goals and CompTIA's Linux+ exam objectives to the pages in this book that cover each topic. This part also includes an extensive Glossary with more than 500 entries plus the JumpStart index, the File Tree index, the Utility index, and the comprehensive Main index.

Supplements

The author's home page (www.sobell.com) contains downloadable listings of the longer programs from this book, as well as pointers to many interesting and useful Linux sites on the World Wide Web, a list of corrections to the book, answers to even-numbered exercises, and a solicitation for corrections, comments, and suggestions.

Thanks

First and foremost, I want to thank Mark L. Taub, Editor-in-Chief, Prentice Hall, who provided encouragement and support through the hard parts of this project. Mark is unique in my 30+ years of book writing experience: an editor who works with the tools I write about. Because Mark runs Linux on his home computer, we shared experiences as I wrote this book. Mark, your comments and direction are invaluable; this book would not exist without your help. Thank you, Mark T.

The production people at Prentice Hall are wonderful to work with: Julie Nahil, Full-Service Production Manager, worked with me day by day during production of this book providing help and keeping everything on track, while John Fuller, Managing Editor, kept the large view in focus. Thanks to Stephanie Geels, Copyeditor, and Andrea Fox, Proofreader, who made each page sparkle and found the mistakes the author left behind.

Thanks also to the folks at Prentice Hall who helped bring this book to life, especially Kim Boedigheimer, Associate Editor, who attended to the many details involved in publishing this book; Heather Fox, Publicist; Stephane Nakib, Marketing Manager; Dan Scherf, Media Developer; Sandra Schroeder, Design Manager; Chuti Prasertsith, Cover Designer; and everyone else who worked behind the scenes to make this book come into being.

I am also indebted to Denis Howe, Editor of *The Free On-Line Dictionary of Computing* (FOLDOC). Denis has graciously permitted me to use entries from his compilation. Be sure to visit www.foldoc.org to look at this dictionary.

A big “thank you” to the folks who read through the drafts of the book and made comments that caused me to refocus parts of the book where things were not clear or were left out altogether:

Liz Joseph, Linux Systems Administrator for HP on the OpenStack project, Linux Systems Administrator, Ubuntu project Member since 2007, and member of the Board of Directors for Partimus.org, had a big impact on this book. She reviewed drafts of many chapters, providing insights, tips, important links, and corrections throughout. I was very impressed with the depth and breadth of her knowledge of Ubuntu.

Nathan Handler, Ubuntu/Debian GNU/Linux Developer; Milo Casagrande, Ubuntu Community; Ray Perlner, NIST (National Institute of Standards and Technology); Robert P. J. Day, Crash Course; Ben Whaley, professional system administrator; and Max Sobell, Carve Systems, provided invaluable input.

Thanks also to the following people who helped with my previous Linux books, which provided a foundation for this book:

Nicklos See, Instructor, Bismarck State College; Eugene Murray, Chair, School of Information Technology, ITT Technical Institute; Doug Hughes, D. E. Shaw Research, LLC; Carl Draper; Rahul Sundaram, Fedora contributor; Tony Godfrey, ITT Technical

Institute, Strongsville, Ohio; Susan Lauber, Lauber System Solutions, Inc.; Sam Sobell, Student, UCLA; Ankur Sinha, Fedora project volunteer; Professor Maria Ripoll; David A. Lane, long-time Linux user and systems architect; Pete Travis, Fedora Documentation Project; Rickard Körkkö, Orc Group; Jennifer Davis, Senior Production Engineer at Yahoo; Matthew Miller, Fedora Project at Red Hat, Inc.; Jiri Popelka, Red Hat, Inc.; Thomas Woerner, Red Hat, Inc.

Richard Woodbury, Site Reliability Engineer, Google, whose knowledge of IPv6 proved invaluable; Lennart Poettering, Red Hat, Inc.; George Vish II, Senior Education Consultant, Hewlett-Packard; Garth Snyder; Nathan Handler; Dick Seabrook, Professor Emeritus, Anne Arundel Community College; Chris Karr, Audacious Software; Scott McCrea, Instructor, ITT Technical Schools; John Dong, Ubuntu Developer, Forums Council Member; Andy Lester, author of *Land the Tech Job You Love: Why Skill and Luck Are Not Enough;* Scott James Remnant, Ubuntu Development Manager and Desktop Team Leader; David Chisnall, Swansea University; Scott Mann, Aztek Networks; Thomas Achtemichuk, Mansueto Ventures; Daniel R. Arfsten, Pro/Engineer Drafter/Designer; Chris Cooper, Senior Education Consultant, Hewlett-Packard Education Services; Sameer Verma, Associate Professor of Information Systems, San Francisco State University; Valerie Chau, Palomar College and Programmers Guild; James Kratzer; Sean McAllister; Nathan Eckenrode, New York Ubuntu Local Community Team; Christer Edwards; Nicolas Merline; Michael Price; Mike Basinger, Ubuntu Community and Forums Council Member; Joe Barker, Ubuntu Forums Staff Member; James Stockford, Systemateka, Inc.; Stephanie Troeth, Book Oven; Doug Sheppard; Bryan Helvey, IT Director, OpenGeoSolutions; and Vann Scott, Baker College of Flint.

Also, thanks to Jesse Keating, Fedora Project; Carsten Pfeiffer, Software Engineer and KDE Developer; Aaron Weber, Ximian; Cristof Falk, Software Developer at CritterDesign; Steve Elgersma, Computer Science Department, Princeton University; Scott Dier, University of Minnesota; Robert Haskins, Computer Net Works; Lars Kellogg-Stedman, Harvard University; Jim A. Lola, Principal Systems Consultant, Privateer Systems; Eric S. Raymond, Cofounder, Open Source Initiative; Scott Mann; Randall Lechlitner, Independent Computer Consultant; Jason Wertz, Computer Science Instructor, Montgomery County Community College; Justin Howell, Solano Community College; Ed Sawicki, The Accelerated Learning Center; David Mercer; Jeffrey Bianchine, Advocate, Author, Journalist; John Kennedy; and Jim Dennis, Starshine Technical Services.

Thanks also to Dustin Puryear, Puryear Information Technology; Gabor Liptak, Independent Consultant; Bart Schaefer, Chief Technical Officer, iPost; Michael J. Jordan, Web Developer, Linux Online; Steven Gibson, Owner, SuperAnt.com; John Viega, Founder and Chief Scientist, Secure Software; K. Rachael Treu, Internet Security Analyst, Global Crossing; Kara Pritchard, K & S Pritchard Enterprises; Glen Wiley, Capital One Finances; Karel Baloun, Senior Software Engineer, Looksmart; Matthew Whitworth; Dameon D. Welch-Abernathy, Nokia Systems; Josh Simon, Consultant; Stan Isaacs; and Dr. Eric H. Herrin II, Vice President, Herrin Software Development.

More thanks go to consultants Lorraine Callahan and Steve Wampler; Ronald Hiller, Graburn Technology; Charles A. Plater, Wayne State University; Bob Palowoda; Tom Bialaski, Sun Microsystems; Roger Hartmuller, TIS Labs at Network Associates; Kaowen Liu; Andy Spitzer; Rik Schneider; Jesse St. Laurent; Steve Bellenot; Ray W. Hiltbrand; Jennifer Witham; Gert-Jan Hagenaars; and Casper Dik.

A Practical Guide to Ubuntu Linux®, Fourth Edition, is based in part on two of my previous UNIX books: *UNIX System V: A Practical Guide* and *A Practical Guide to the UNIX System*. Many people helped me with those books, and thanks here go to Pat Parseghian; Dr. Kathleen Hemenway; Brian LaRose; Byron A. Jeff, Clark Atlanta University; Charles Stross; Jeff Gitlin, Lucent Technologies; Kurt Hockenbury; Maury Bach, Intel Israel; Peter H. Salus; Rahul Dave, University of Pennsylvania; Sean Walton, Intelligent Algorithmic Solutions; Tim Segall, Computer Sciences Corporation; Behrouz Forouzan, DeAnza College; Mike Keenan, Virginia Polytechnic Institute and State University; Mike Johnson, Oregon State University; Jandelyn Plane, University of Maryland; Arnold Robbins and Sathis Menon, Georgia Institute of Technology; Cliff Shaffer, Virginia Polytechnic Institute and State University; and Steven Stepanek, California State University, Northridge, for reviewing the book.

I continue to be grateful to the many people who helped with the early editions of my UNIX books. Special thanks are due to Roger Sippl, Laura King, and Roy Harrington for introducing me to the UNIX system. My mother, Dr. Helen Sobell, provided invaluable comments on the original manuscript at several junctures. Also, thanks go to Isaac Rabinovitch, Professor Raphael Finkel, Professor Randolph Bentson, Bob Greenberg, Professor Udo Pooch, Judy Ross, Dr. Robert Veroff, Dr. Mike Denny, Joe DiMartino, Dr. John Mashey, Diane Schulz, Robert Jung, Charles Whitaker, Don Cragun, Brian Dougherty, Dr. Robert Fish, Guy Harris, Ping Liao, Gary Lindgren, Dr. Jarrett Rosenberg, Dr. Peter Smith, Bill Weber, Mike Bianchi, Scooter Morris, Clarke Echols, Oliver Grillmeyer, Dr. David Korn, Dr. Scott Weikart, and Dr. Richard Curtis.

I take responsibility for any errors and omissions in this book. If you find one or just have a comment, let me know (mgs@sobell.com) and I will fix it in the next printing. My home page (www.sobell.com) offers copies of the longer scripts from the book and pointers to interesting Linux pages on the Internet. You can follow me at twitter.com/marksobell.

Mark G. Sobell
San Francisco, California

This page intentionally left blank

This page intentionally left blank

5

The Shell

In This Chapter

The Working Directory. 151
Your Home Directory 151
The Command Line 152
Standard Input and
Standard Output 159
Redirection 161
Pipelines 166
Running a Command in the
Background 171
kill: Aborting a Background Job . . 172
Filename Generation/Pathname
Expansion 173
Builtins 178

Objectives

After reading this chapter you should be able to:

- List special characters and methods of preventing the shell from interpreting these characters
- Describe a simple command
- Understand command-line syntax and run commands that include options and arguments
- Explain how the shell interprets the command line
- Redirect output of a command to a file, overwriting the file or appending to it
- Redirect input for a command so it comes from a file
- Connect commands using a pipeline
- Run commands in the background
- Use special characters as wildcards to generate filenames
- Explain the difference between a stand-alone utility and a shell builtin

The introduction to the command line on page 125 described some of the advantages of using the command line over a GUI, how to use a terminal emulator, how to correct mistakes on the command line, and how to run some command-line utilities. This chapter takes a close look at the shell and explains how to use some of its features. It discusses command-line syntax and describes how the shell processes a command line and initiates execution of a program. This chapter also explains how to redirect input to and output from a command, construct pipelines and filters on the command line, and run a command in the background. The final section covers filename expansion and explains how you can use this feature in your everyday work.

The exact wording of the shell output differs from shell to shell: What the shell you are using displays might differ slightly from what appears in this book. Refer to Chapter 9 for more information on bash (the default shell under Ubuntu) and to Chapter 28 for information on writing and executing bash shell scripts.

LE Special Characters

Special characters, which have a special meaning to the shell, are discussed in "Filename Generation/Pathname Expansion" on page 173. These characters are mentioned here so you can avoid accidentally using them as regular characters until you understand how the shell interprets them. Avoid using any of the following characters in a filename (even though emacs and some other programs do) because they make the file harder to reference on the command line:

```
& ; | * ? ' " ` [ ] ( ) $ < > { } # / \ ! ~
```

Whitespace Although not considered special characters, RETURN, SPACE, and TAB have special meanings to the shell. RETURN usually ends a command line and initiates execution of a command. The SPACE and TAB characters separate tokens (elements) on the command line and are collectively known as *whitespace* or *blanks.*

Quoting special characters If you need to use a character that has a special meaning to the shell as a regular character, you can *quote* (or *escape*) it. When you quote a special character, you prevent the shell from giving it special meaning. The shell treats a quoted special character as a regular character. However, a slash (/) is always a separator in a pathname, even when you quote it.

Backslash To quote a character, precede it with a backslash (\). When two or more special characters appear together, you must precede each with a backslash (e.g., you would enter ** as **). You can quote a backslash just as you would quote any other special character—by preceding it with a backslash (\\).

Single quotation marks Another way of quoting special characters is to enclose them between single quotation marks: '**'. You can quote many special and regular characters between a pair of single quotation marks: **'This is a special character: >'**. The regular characters are interpreted as usual, and the shell also interprets the special characters as regular characters.

The only way to quote the erase character (CONTROL-H), the line kill character (CONTROL-U), and other control characters (try CONTROL-M) is by preceding each with a CONTROL-V. Single quotation marks and backslashes do not work. Try the following:

```
$ echo 'xxxxxxCONTROL-U'
$ echo xxxxxxCONTROL-V CONTROL-U
```

optional Although you cannot see the CONTROL-U displayed by the second of the preceding pair of commands, it is there. The following command sends the output of echo (page 227) through a pipeline (page 166) to od (octal display; see the od man page) to display CONTROL-U as octal 25 (025):

```
$ echo xxxxxxCONTROL-V CONTROL-U | od -c
0000000   x   x   x   x   x   x 025  \n
0000010
```

The **\n** is the NEWLINE character that echo sends at the end of its output.

Ordinary Files and Directory Files

Ordinary files, or simply *files,* are files that can hold documents, pictures, programs, and other kinds of data. *Directory files,* also referred to as *directories* or *folders,* can hold ordinary files and other directory files. For more information refer to "Ordinary Files and Directory Files" on page 185.

The Working Directory

LPI pwd While you are logged in on a character-based interface to a Linux system, you are always associated with a directory. The directory you are associated with is called the *working directory* or *current directory*. Sometimes this association is referred to in a physical sense: "You are *in* (or *working in*) the **zach** directory." The pwd (print working directory) builtin displays the pathname of the working directory.

```
login: max
Password:
Last login: Wed Oct 20 11:14:21 from 172.16.192.150
$ pwd
/home/max
```

LE Your Home Directory

When you first log in on a Linux system or start a terminal emulator window, the working directory is your *home directory*. To display the pathname of your home directory, use pwd just after you log in.

LE The Command Line

Command This book uses the term *command* to refer to both the characters you type on the command line and the program that action invokes.

Command line A *command line* comprises a simple command (below), a pipeline (page 166), or a list (page 170).

A Simple Command

The shell executes a program when you enter a command in response to its prompt. For example, when you give an ls command, the shell executes the utility program named ls. You can cause the shell to execute other types of programs—such as shell scripts, application programs, and programs you have written—in the same way. The line that contains the command, including any arguments, is called a *simple command*. The following sections discuss simple commands; see page 155 for a more technical and complete description of a simple command.

Syntax

Command-line syntax dictates the ordering and separation of the elements on a command line. When you press the RETURN key after entering a command, the shell scans the command line for proper syntax. The syntax for a simple command is

***command** [arg1] [arg2] ... [argn]* RETURN

Whitespace (any combination of SPACEs and/or TABs) must separate elements on the command line. The ***command*** is the name of the command, ***arg1*** through ***argn*** are arguments, and RETURN is the keystroke that terminates the command line. The brackets in the command-line syntax indicate that the arguments they enclose are optional. Not all commands require arguments: Some commands do not allow arguments; other commands allow a variable number of arguments; and still others require a specific number of arguments. Options, a special kind of argument, are usually preceded by one or two hyphens (–).

Command Name

Usage message Some useful Linux command lines consist of only the name of the command without any arguments. For example, ls by itself lists the contents of the working directory. Commands that require arguments typically give a short error message, called a *usage message,* when you use them without arguments, with incorrect arguments, or with the wrong number of arguments.

For example, the mkdir (make directory) utility requires an argument that specifies the name of the directory you want it to create. Without this argument, it displays a usage message (*operand* is another term for "argument"):

```
$ mkdir
mkdir: missing operand
Try 'mkdir --help' for more information.
```

LE Arguments

Token On the command line each sequence of nonblank characters is called a *token* or *word*. An *argument* is a token that a command acts on (e.g., a filename, a string of characters, a number). For example, the argument to a vim or emacs command is the name of the file you want to edit.

The following command line uses cp to copy the file named **temp** to **tempcopy**:

```
$ cp temp tempcopy
```

Arguments are numbered starting with the command itself, which is argument zero. In this example, **cp** is argument zero, **temp** is argument one, and **tempcopy** is argument two. The cp utility requires at least two arguments on the command line. Argument one is the name of an existing file. In this case, argument two is the name of the file that cp is creating or overwriting. Here the arguments are not optional; both arguments must be present for the command to work. When you do not supply the right number or kind of arguments, cp displays a usage message. Try typing **cp** and then pressing RETURN.

LE Options

An *option* is an argument that modifies the effects of a command. These arguments are called options because they are usually optional. You can frequently specify more than one option, modifying the command in several ways. Options are specific to and interpreted by the program that the command line calls, not the shell.

By convention, options are separate arguments that follow the name of the command and usually precede other arguments, such as filenames. Many utilities require options to be prefixed with a single hyphen. However, this requirement is specific to the utility and not the shell. GNU long (multicharacter) program options are frequently prefixed with two hyphens. For example, **––help** generates a (sometimes extensive) usage message.

The first of the following commands shows the output of an ls command without any options. By default, ls lists the contents of the working directory in alphabetical order, vertically sorted in columns. Next the **–r** (reverse order; because this is a GNU utility, you can also specify **––reverse**) option causes the ls utility to display the list of files in reverse alphabetical order, still sorted in columns. The **–x** option causes ls to display the list of files in horizontally sorted rows.

```
$ ls
hold   mark  names   oldstuff  temp  zach
house  max   office  personal  test
$ ls -r
zach  temp      oldstuff  names  mark   hold
test  personal  office    max    house
$ ls -x
hold      house     mark  max   names  office
oldstuff  personal  temp  test  zach
```

Combining options

When you need to use several options, you can usually group multiple single-letter options into one argument that starts with a single hyphen; do not put SPACEs between the options. You cannot combine options that are preceded by two hyphens in this way. Specific rules for combining options depend on the program you are running. The next example shows both the **–r** and **–x** options with the ls utility. Together these options generate a list of filenames in horizontally sorted rows in reverse alphabetical order.

```
$ ls -rx
zach   test  temp  personal  oldstuff  office
names  max   mark  house     hold
```

Most utilities allow you to list options in any order; thus **ls –xr** produces the same results as **ls –rx**. The command **ls –x –r** also generates the same list.

The ––help option

tip

Many utilities display a (sometimes extensive) help message when you call them with an argument of **––help**. All utilities developed by the GNU Project (page 3) accept this option. Following is the help message displayed by the bzip2 compression utility (page 253).

```
$ bzip2 --help
bzip2, a block-sorting file compressor.  Version 1.0.6, 6-Sept-2010.

   usage: bunzip2 [flags and input files in any order]

   -h --help           print this message
   -d --decompress     force decompression
   -z --compress       force compression
   -k --keep           keep (don't delete) input files
   -f --force          overwrite existing output files
...
   If invoked as 'bzip2', default action is to compress.
              as 'bunzip2',  default action is to decompress.
              as 'bzcat', default action is to decompress to stdout.
...
```

Option arguments

Some utilities have options that require arguments. These arguments are not optional. For example, the gcc utility (C compiler) has a **–o** (output) option that must be followed by the name you want to give the executable file that gcc generates. Typically an argument to an option is separated from its option letter by a SPACE:

```
$ gcc -o prog prog.c
```

Some utilities sometimes require an equal sign between an option and its argument. For example, you can specify the width of output from diff in two ways:

```
$ diff -W 60 filea fileb
```

or

```
$ diff --width=60 filea fileb
```

Arguments that start with a hyphen

Another convention allows utilities to work with arguments, such as filenames, that start with a hyphen. If a file named **–l** is in the working directory, the following command is ambiguous:

```
$ ls -l
```

This command could be a request to display a long listing of all files in the working directory (**–l** option) or a request for a listing of the file named **–l**. The ls utility interprets it as the former. Avoid creating a file whose name begins with a hyphen. If you do create such a file, many utilities follow the convention that a **––** argument (two consecutive hyphens) indicates the end of the options (and the beginning of the arguments). To disambiguate the preceding command, you can type

```
$ ls -- -l
```

Using two consecutive hyphens to indicate the end of the options is a convention, not a hard-and-fast rule, and a number of utilities do not follow it (e.g., find). Following this convention makes it easier for users to work with a program you write.

Displaying readable file sizes: the –h option

tip Most utilities that report on file sizes specify the size of a file in bytes. Bytes work well when you are dealing with smaller files, but the numbers can be difficult to read when you are working with file sizes that are measured in gigabytes or terabytes. Use the **–h** (or **––human-readable**) option to display file sizes in kilobytes, megabytes, gigabytes, and terabytes. Experiment with the **df –h** (disk free) and **ls –lh** commands.

For utilities that do not follow this convention, there are other ways to specify a filename that begins with a hyphen. You can use a period to refer to the working directory and a slash to indicate the following filename refers to a file in the working directory:

```
$ ls ./-l
```

You can also specify the absolute pathname of the file:

```
$ ls /home/max/-l
```

optional

Simple Commands

This section expands on the discussion of command-line syntax that starts on page 152.

A simple command comprises zero or more variable assignments followed by a command line. It is terminated by a control operator (e.g., &, ;, |, NEWLINE; page 347). A simple command has the following syntax:

[name=value ...] ***command-line***

The shell assigns a ***value*** to each ***name*** and places it in the environment (page 1054) of the program that ***command-line*** calls so it is available to the called program and its children as a variable. The shell evaluates the ***name=value*** pairs from left to right, so if ***name*** appears more than once in this list, the rightmost ***value*** takes precedence.

The ***command-line*** might include redirection operators such as **>** and **<** (page 161). The exit status (page 1051) of a simple command is its return value.

Placing a variable in the environment of a child

The following commands demonstrate how you can assign a value to a name (variable) and place that name in the environment of a child program; the variable is not available to the interactive shell you are running (the parent program). The script named **echo_ee** displays the value of the variable named **ee**. The first call to **echo_ee** shows **ee** is not set in the child shell running the script. When the call to **echo_ee** is preceded by assigning a value to **ee**, the script displays the value of **ee** in the child shell. The final command shows **ee** has not been set in the interactive shell.

```
$ cat echo_ee
echo "The value of the ee variable is: $ee"

$ ./echo_ee
The value of the ee variable is:
$ ee=88 ./echo_ee
The value of the ee variable is: 88
$ echo $ee

$
```

Processing the Command Line

As you enter a command line, the tty device driver (part of the Linux kernel) examines each character to see whether it must take immediate action. When you press CONTROL-H (to erase a character) or CONTROL-U (to kill a line), the device driver immediately adjusts the command line as required; the shell never sees the character(s) you erased or the line you killed. Often a similar adjustment occurs when you press CONTROL-W (to erase a word). When the character you entered does not require immediate action, the device driver stores the character in a buffer and waits for additional characters. When you press RETURN, the device driver passes the command line to the shell for processing.

Parsing the command line

When the shell processes a command line, it looks at the line as a whole and *parses* (breaks) it into its component parts (Figure 5-1). Next the shell looks for the name of the command. Usually the name of the command is the first item on the command line after the prompt (argument zero). The shell takes the first characters on the command line up to the first blank (TAB or SPACE) and then looks for a command with that name. The command name (the first token) can be specified on the command line either as a simple filename or as a pathname. For example, you can call the ls command in either of the following ways:

```
$ ls
```

or

```
$ /bin/ls
```

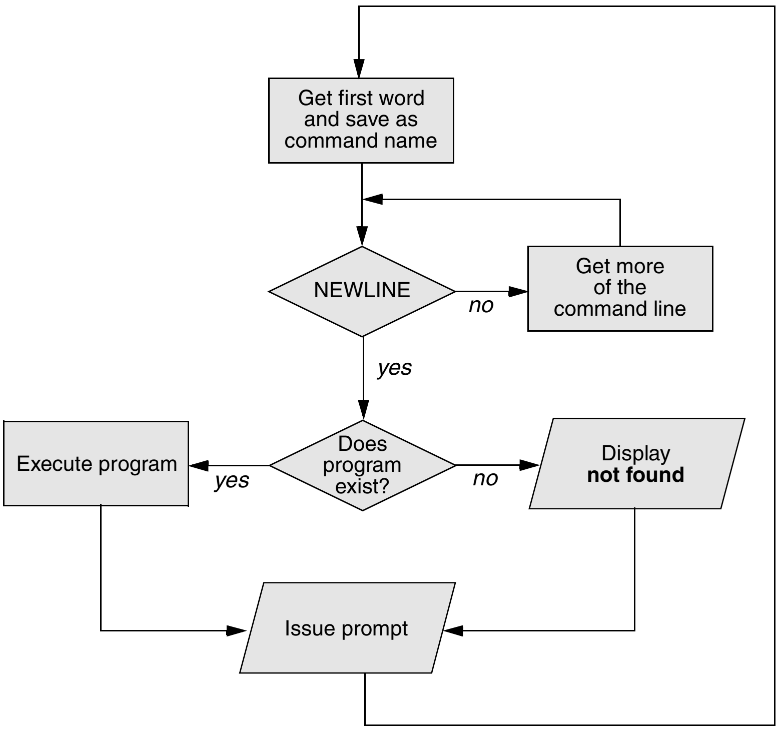

Figure 5-1 Processing the command line

optional The shell does not require the name of the program to appear first on the command line. Thus you can structure a command line as follows:

```
$ >bb <aa cat
```

This command runs cat with standard input coming from the file named **aa** and standard output going to the file named **bb**. When the shell recognizes the redirect symbols (page 161), it processes them and their arguments before finding the name of the program that the command line is calling. This is a properly structured—albeit rarely encountered and possibly confusing—command line.

LPI Absolute versus relative pathnames From the command line, there are three ways you can specify the name of a file you want the shell to execute: as an absolute pathname (starts with a slash [/]; page 189), as a relative pathname (includes a slash but does not start with a slash; page 190), or as a simple filename (no slash). When you specify the name of a file for the shell to execute in either of the first two ways (the pathname includes a slash), the shell looks in the specified directory for a file with the specified name that you have permission to execute. When you specify a simple filename (no slash), the shell searches through a list of directories for a filename that matches the specified name and for which you have execute permission. The shell does not look through all directories but only the ones specified by the variable named **PATH**. Refer to page 365 for more information on **PATH**. Also refer to the discussion of the which and whereis utilities on page 263.

When it cannot find the file, bash displays the following message:

```
$ abc
bash: abc: command not found...
```

Some systems are set up to suggest where you might be able to find the program you tried to run. One reason the shell might not be able to find the executable file is that it is not in a directory listed in the **PATH** variable. Under bash the following command temporarily adds the working directory (.) to **PATH**:

```
$ PATH=$PATH:.
```

For security reasons, it is poor practice to add the working directory to **PATH** permanently; see the following tip and the one on page 366.

When the shell finds the file but cannot execute it (i.e., because you do not have execute permission for the file), it displays a message similar to

```
$ def
bash: ./def: Permission denied
```

See "**ls –l**: Displays Permissions" on page 199 for information on displaying access permissions for a file and "chmod: Changes Access Permissions" on page 201 for instructions on how to change file access permissions.

Try giving a command as ./*command*

tip You can always execute an executable file in the working directory by prepending **./** to the name of the file. Because **./filename** is a relative pathname, the shell does not consult **PATH** when looking for **filename**. For example, if **myprog** is an executable file in the working directory, you can execute it using the following command (regardless of how **PATH** is set):

```
$ ./myprog
```

Executing a Command

LE Process If it finds an executable file with the name specified on the command line, the shell starts a new process. A *process* is the execution of a command by Linux (page 379). The shell makes each command-line argument, including options and the name of the command, available to the called program. While the command is executing, the shell waits for the process to finish. At this point the shell is in an inactive state named *sleep*. When the program finishes execution, it passes its exit status (page 1051) to the shell. The shell then returns to an active state (wakes up), issues a prompt, and waits for another command.

The shell does not process arguments Because the shell does not process command-line arguments but merely passes them to the called program, the shell has no way of knowing whether a particular option or other argument is valid for a given program. Any error or usage messages about options or arguments come from the program itself. Some utilities ignore bad options.

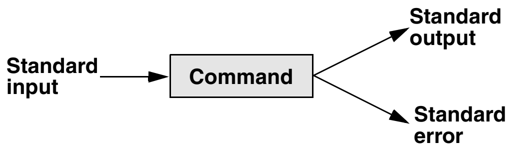

Figure 5-2 The command does not know where standard input comes from or where standard output and standard error go

Editing the Command Line

You can repeat and edit previous commands and edit the current command line. See pages 131 and 384 for more information.

Standard Input and Standard Output

Standard output is a place to which a program can send information (e.g., text). The program never "knows" where the information it sends to standard output is going (Figure 5-2). The information can go to a printer, an ordinary file, or the screen. The following sections show that by default the shell directs standard output from a command to the screen[1] and describe how you can cause the shell to redirect this output to another file.

Standard input is a place a program gets information from; by default, the shell directs standard input from the keyboard. As with standard output the program never "knows" where the information comes from. The following sections explain how to redirect standard input to a command so it comes from an ordinary file instead of from the keyboard.

In addition to standard input and standard output, a running program has a place to send error messages: *standard error*. By default, the shell directs standard error to the screen. Refer to page 339 for more information on redirecting standard error.

optional By convention, a process expects that the program that called it (frequently the shell) has set up standard input, standard output, and standard error so the process can use them immediately. The called process does not *have* to know which files or devices are connected to standard input, standard output, or standard error.

However, a process *can* query the kernel to get information about the device that standard input, standard output, or standard error is connected to. For example, the ls utility displays output in multiple columns when the output goes to the screen, but generates a single column of output when the output is redirected to a file or another

1. This book uses the term *screen* to refer to a screen, terminal emulator window, or workstation—in other words, to the device that the shell displays its prompt and messages on.

program. The ls utility uses the **isatty**() system call to determine whether output is going to the screen (a tty). In addition, ls can use another system call to determine the width of the screen it is sending output to; with this information it can modify its output to fit the screen. Compare the output of ls by itself and when you send it through a pipeline to less. See page 1042 for information on how you can determine whether standard input and standard output of shell scripts is going to/coming from the terminal.

The Screen as a File

LPI Device file Chapter 6 discusses ordinary files, directory files, and hard and soft links. Linux has an additional type of file: a *device file*. A device file resides in the file structure, usually in the **/dev** directory, and represents a peripheral device, such as a terminal, printer, or disk drive.

The device name the who utility displays following a username is the filename of the terminal that user is working on. For example, when who displays the device name **pts/4**, the pathname of the terminal is **/dev/pts/4**. When you work with multiple windows, each window has its own device name. You can also use the tty utility to display the name of the device that you give the command from. Although you would not normally have occasion to do so, you can read from and write to this file as though it were a text file. Reading from the device file that represents the terminal you are using reads what you enter on the keyboard; writing to it displays what you write on the screen.

The Keyboard and Screen as Standard Input and Standard Output

After you log in, the shell directs standard output of commands you enter to the device file that represents the terminal (Figure 5-3). Directing output in this manner causes it to appear on the screen. The shell also directs standard input to come from the same file, so commands receive as input anything you type on the keyboard.

LPI cat The cat utility provides a good example of the way the keyboard and screen function as standard input and standard output, respectively. When you run cat, it copies a file to standard output. Because the shell directs standard output to the screen, cat displays the file on the screen.

Up to this point cat has taken its input from the filename (argument) you specify on the command line. When you do not give cat an argument (i.e., when you give the command cat followed immediately by RETURN), cat takes its input from standard input. Thus, when called without an argument, cat copies standard input to standard output, one line at a time.

To see how cat works, type **cat** and press RETURN in response to the shell prompt. Nothing happens. Enter a line of text and press RETURN. The same line appears just under the one

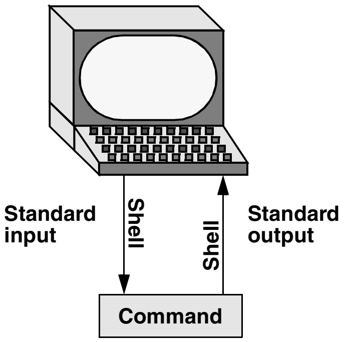

Figure 5-3 By default, standard input comes from the keyboard, and standard output goes to the screen

you entered. The cat utility is working. Because the shell associates cat's standard input with the keyboard and cat's standard output with the screen, when you type a line of text cat copies the text from standard input (the keyboard) to standard output (the screen). The next example shows this exchange.

```
$ cat
This is a line of text.
This is a line of text.
Cat keeps copying lines of text
Cat keeps copying lines of text
until you press CONTROL-D at the beginning
until you press CONTROL-D at the beginning
of a line.
of a line.
CONTROL-D
$
```

CONTROL-D signals EOF

The cat utility keeps copying text until you enter CONTROL-D on a line by itself. Pressing CONTROL-D causes the tty device driver to send an EOF (end of file) signal to cat. This signal indicates to cat that it has reached the end of standard input and there is no more text for it to copy. The cat utility then finishes execution and returns control to the shell, which displays a prompt.

LE+ Redirection

The term *redirection* encompasses the various ways you can cause the shell to alter where standard input of a command comes from and where standard output goes to. By default, the shell associates standard input and standard output of a command with the keyboard and the screen. You can cause the shell to redirect standard input or standard output of any command by associating the input or output with a command or file other than the device file representing the keyboard or the screen. This section demonstrates how to redirect input/output from/to text files and utilities.

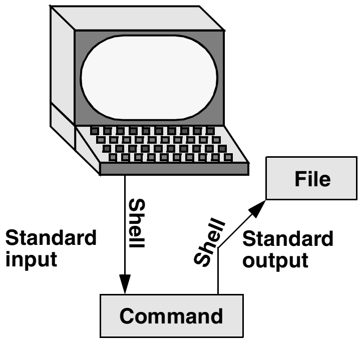

Figure 5-4 Redirecting standard output

LPI Redirecting Standard Output

The *redirect output symbol* (**>**) instructs the shell to redirect the output of a command to the specified file instead of to the screen (Figure 5-4). The syntax of a command line that redirects output is

command** [arguments]* > ***filename

where ***command*** is any executable program (e.g., an application program or a utility), ***arguments*** are optional arguments, and ***filename*** is the name of the ordinary file the shell redirects the output to.

Using cat to create a file

The next example uses cat to demonstrate output redirection. This example contrasts with the example on page 161, where standard input *and* standard output are associated with the keyboard and screen. The input in the following example comes from the keyboard. The redirect output symbol on the command line causes the shell to associate cat's standard output with the **sample.txt** file specified following this symbol.

```
$ cat > sample.txt
This text is being entered at the keyboard and
cat is copying it to a file.
Press CONTROL-D to indicate the
end of file.
CONTROL-D
$
```

Redirecting output can destroy a file I

caution Use caution when you redirect output to a file. If the file exists, the shell will overwrite it and destroy its contents. For more information see the tip "Redirecting output can destroy a file II" on page 164.

After giving the command and typing the text shown in the previous example, the **sample.txt** file contains the text you entered. You can use cat with an argument of **sample.txt** to display this file. The next section shows another way to use cat to display the file.

The previous example shows that redirecting standard output from cat is a handy way to create a file without using an editor. The drawback is that once you enter a line and press RETURN, you cannot edit the text until after you finish creating the file. While you are entering a line, the erase and kill keys work to delete text on that line. This procedure is useful for creating short, simple files.

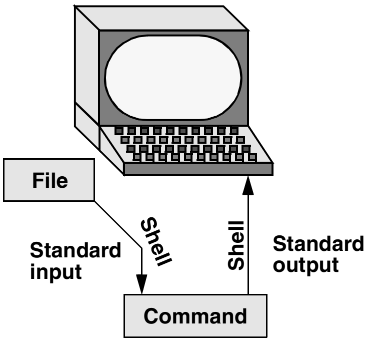

Figure 5-5 Redirecting standard input

The next example shows how to run cat and use the redirect output symbol to *catenate* (join one after the other—the derivation of the name of the cat utility) several files into one larger file. The first three commands display the contents of three files: **stationery**, **tape**, and **pens**. The next command shows cat with three filenames as arguments. When you call it with more than one filename, cat copies the files, one at a time, to standard output. This command redirects standard output to the file named **supply_orders**. The final cat command shows that **supply_orders** contains the contents of the three original files.

```
$ cat stationery
2,000 sheets letterhead ordered:   October 7
$ cat tape
1 box masking tape ordered:        October 14
5 boxes filament tape ordered:     October 28
$ cat pens
12 doz. black pens ordered:        October 4

$ cat stationery tape pens > supply_orders

$ cat supply_orders
2,000 sheets letterhead ordered:   October 7
1 box masking tape ordered:        October 14
5 boxes filament tape ordered:     October 28
12 doz. black pens ordered:        October 4
```

LPI Redirecting Standard Input

Just as you can redirect standard output, so you can redirect standard input. The *redirect input symbol* (**<**) instructs the shell to redirect a command's input to come from the specified file instead of from the keyboard (Figure 5-5). The syntax of a command line that redirects input is

command** [**arguments**] < **filename

where ***command*** is any executable program (such as an application program or a utility), ***arguments*** are optional arguments, and ***filename*** is the name of the ordinary file the shell redirects the input from.

The next example shows cat with its input redirected from the **supply_orders** file created in the previous example and standard output going to the screen. This setup causes cat to display the **supply_orders** file on the screen. The system automatically supplies an EOF signal at the end of an ordinary file.

```
$ cat < supply_orders
2,000 sheets letterhead ordered:   October 7
1 box masking tape ordered:        October 14
5 boxes filament tape ordered:     October 28
12 doz. black pens ordered:        October 4
```

Utilities that take input from a file or standard input

Giving a cat command with input redirected from a file yields the same result as giving a cat command with the filename as an argument. The cat utility is a member of a class of utilities that function in this manner. Other members of this class of utilities include lpr, sort, grep, and Perl. These utilities first examine the command line that called them. If the command line includes a filename as an argument, the utility takes its input from the specified file. If no filename argument is present, the utility takes its input from standard input. It is the utility or program—not the shell or operating system—that functions in this manner.

Redirecting output can destroy a file II

caution

Depending on which shell you are using and how the environment is set up, a command such as the following can yield undesired results:

```
$ cat orange pear > orange
cat: orange: input file is output file
```

Although cat displays an error message, the shell destroys the contents of the existing **orange** file. The new **orange** file will have the same contents as **pear** because the first action the shell takes when it sees the redirection symbol (**>**) is to remove the contents of the original **orange** file. If you want to catenate two files into one, use cat to put the two files into a temporary file and then use mv to rename the temporary file:

```
$ cat orange pear > temp
$ mv temp orange
```

What happens in the next example can be even worse. The user giving the command wants to search through files **a**, **b**, and **c** for the word **apple** and redirect the output from grep (page 240) to the file **a.output**. Unfortunately the user enters the filename as **a output**, omitting the period and inserting a SPACE in its place:

```
$ grep apple a b c > a output
grep: output: No such file or directory
```

The shell obediently removes the contents of **a** and then calls grep. The error message could take a moment to appear, giving you a sense that the command is running correctly. Even after you see the error message, it might take a while to realize that you have destroyed the contents of **a**.

noclobber: Prevents Overwriting Files

The shell provides the **noclobber** feature, which prevents you from overwriting a file using redirection. Enable this feature by setting **noclobber** using the command **set –o noclobber**. The same command with **+o** unsets **noclobber**. With **noclobber** set, if you redirect output to an existing file, the shell displays an error message and does not execute the command. The following example creates a file using touch, sets **noclobber**, attempts to redirect the output from echo to the newly created file, unsets **noclobber**, and performs the same redirection:

```
$ touch tmp
$ set -o noclobber
$ echo "hi there" > tmp
-bash: tmp: cannot overwrite existing file
$ set +o noclobber
$ echo "hi there" > tmp
```

You can override **noclobber** by putting a pipeline symbol after the redirect symbol (**>|**). In the following example, the user creates a file by redirecting the output of date. Next the user sets the **noclobber** variable and redirects output to the same file again. The shell displays an error message. Then the user places a pipeline symbol after the redirect symbol, and the shell allows the user to overwrite the file.

```
$ date > tmp2
$ set -o noclobber
$ date > tmp2
-bash: tmp2: cannot overwrite existing file
$ date >| tmp2
```

Appending Standard Output to a File

Do not trust noclobber

caution Appending output is simpler than the two-step procedure described in the preceding caution box but you must be careful to include both greater than signs. If you accidentally use only one greater than sign and the **noclobber** feature is not set, the shell will overwrite the **orange** file. Even if you have the **noclobber** feature turned on, it is a good idea to keep backup copies of the files you are manipulating in case you make a mistake.

Although it protects you from overwriting a file using redirection, **noclobber** does not stop you from overwriting a file using cp or mv. These utilities include the **–i** (interactive) option that helps protect you from this type of mistake by verifying your intentions when you try to overwrite a file. For more information see the tip "cp can destroy a file" on page 232.

The *append output symbol* (**>>**) causes the shell to add new information to the end of a file, leaving existing information intact. This symbol provides a convenient way of catenating two files into one. The following commands demonstrate the action of the append output symbol. The second command accomplishes the catenation described in the preceding caution box:

```
$ cat orange
this is orange
$ cat pear >> orange
$ cat orange
this is orange
this is pear
```

The first command displays the contents of the **orange** file. The second command appends the contents of the **pear** file to the **orange** file. The final command displays the result.

The next example shows how to create a file that contains the date and time (the output from date), followed by a list of who is logged in (the output from who). The first command in the example redirects the output from date to the file named **whoson**. Then cat displays the file. The next command appends the output from who to the **whoson** file. Finally cat displays the file containing the output of both utilities.

```
$ date > whoson
$ cat whoson
Wed Mar 27 14:31:18 PST 2013
$ who >> whoson
$ cat whoson
Wed Mar 27 14:31:18 PST 2013
sam        tty1         2013-03-27 05:00(:0)
max        pts/4        2013-03-27 12:23(:0.0)
max        pts/5        2013-03-27 12:33(:0.0)
zach       pts/7        2013-03-26 08:45 (172.16.192.1)
```

/dev/null: Making Data Disappear

The **/dev/null** device is a *data sink,* commonly referred to as a *bit bucket.* You can redirect output you do not want to keep or see to **/dev/null**, and the output will disappear without a trace:

```
$ echo "hi there" > /dev/null
$
```

Reading from **/dev/null** yields a null string. The following command truncates the file named **messages** to zero length while preserving the ownership and permissions of the file:

```
$ ls -lh messages
-rw-rw-r--. 1 sam pubs 125K 03-16 14:30 messages
$ cat /dev/null > messages
$ ls -lh messages
-rw-rw-r--. 1 sam pubs 0 03-16 14:32 messages
```

See also page 481.

LE+ Pipelines

A *pipeline* consists of one or more commands separated by a pipeline symbol (|). The shell connects standard output (and optionally standard error) of the command preceding the pipeline symbol to standard input of the command following the pipeline symbol. A pipeline has the same effect as redirecting standard output of one command to a file and then using that file as standard input to another command. A pipeline does away with separate commands and the intermediate file. The syntax of a pipeline is

command_a [arguments]* | *command_b [arguments]

The preceding command line uses a pipeline to effect the same result as the following three commands:

command_a [arguments] > temp
command_b [arguments] < temp
rm temp

In the preceding sequence of commands, the first line redirects standard output from ***command_a*** to an intermediate file named ***temp***. The second line redirects standard input for ***command_b*** to come from ***temp***. The final line deletes ***temp***. The pipeline syntax is not only easier to type but also is more efficient because it does not create a temporary file.

optional More precisely, a bash pipeline comprises one or more simple commands (page 155) separated by a | or |& control operator. A pipeline has the following syntax:

[time] [!] ***command1*** *[| | |&* ***command2*** *...]*

where *time* is an optional utility that summarizes the system resources used by the pipeline, *!* logically negates the exit status returned by the pipeline, and the ***commands*** are simple commands (page 155) separated by | or |&. The | control operator sends standard output of ***command1*** to standard input of ***command2***. The |& control operator is short for **2>&1 |** (see "Sending errors through a pipeline" on page 341) and sends standard output and standard error of ***command1*** to standard input of ***command2***. The exit status of a pipeline is the exit status of the last simple command unless **pipefail** (page 409) is set, in which case the exit status is the rightmost simple command that failed (returned a nonzero exit status) or zero if all simple commands completed successfully.

Examples of Pipelines

LPI tr You can include in a pipeline any utility that accepts input either from a file specified on the command line or from standard input. You can also include utilities that accept input only from standard input. For example, the tr (translate) utility takes its input from standard input only. In its simplest usage tr has the following syntax:

tr ***string1 string2***

The tr utility accepts input from standard input and looks for characters that match one of the characters in ***string1***. Upon finding a match, it translates the matched character in ***string1*** to the corresponding character in ***string2***. That is, the first character in ***string1*** translates into the first character in ***string2***, and so forth. The tr utility sends its output to standard output. In both of the following tr commands, tr displays the contents of the **abstract** file with the letters **a**, **b**, and **c** translated into **A**, **B**, and **C**, respectively:

```
$ cat abstract
I took a cab today!

$ cat abstract | tr abc ABC
I took A CAB todAy!
$ tr abc ABC < abstract
I took A CAB todAy!
```

The tr utility does not change the contents of the original file; it cannot change the original file because it does not "know" the source of its input.

lpr The lpr (line printer) utility accepts input from either a file or standard input. When you type the name of a file following lpr on the command line, it places that file in the print queue. When you do not specify a filename on the command line, lpr takes input from standard input. This feature enables you to use a pipeline to redirect input to lpr. The first set of the following commands shows how you can use ls and lpr with an intermediate file (**temp**) to send a list of the files in the working directory to the printer. If the **temp** file exists, the first command overwrites its contents. The second set of commands uses a pipeline to send the same list (with the exception of **temp**) to the printer.

```
$ ls > temp
$ lpr temp
$ rm temp
```

or

```
$ ls | lpr
```

sort The commands in next example redirect the output from the who utility to **temp** and then display this file in sorted order. The sort utility (page 247) takes its input from the file specified on the command line or, when a file is not specified, from standard input; it sends its output to standard output. The sort command line takes its input from standard input, which is redirected (**<**) to come from **temp**. The output sort sends to the screen lists the users in sorted (alphabetical) order. Because sort can take its input from standard input or from a file named on the command line, omitting the **<** symbol from the command line yields the same result.

```
$ who > temp
$ sort < temp
max        pts/4        2013-03-24 12:23
max        pts/5        2013-03-24 12:33
sam        tty1         2013-03-24 05:00
zach       pts/7        2013-03-23 08:45
$ rm temp
```

The next example achieves the same result without creating the **temp** file. Using a pipeline, the shell redirects the output from who to the input of sort. The sort utility takes input from standard input because no filename follows it on the command line.

```
$ who | sort
max        pts/4        2013-03-24 12:23
max        pts/5        2013-03-24 12:33
sam        tty1         2013-03-24 05:00
zach       pts/7        2013-03-23 08:45
```

grep When many people are using the system and you want information about only one of them, you can send the output from who to grep (page 240) using a pipeline. The grep utility displays the line containing the string you specify—**sam** in the following example:

```
$ who | grep sam
sam       tty1        2013-03-24 05:00
```

less and more Another way of handling output that is too long to fit on the screen, such as a list of files in a crowded directory, is to use a pipeline to send the output through less or more (both on page 228).

```
$ ls | less
```

The less utility displays text one screen at a time. To view another screen of text, press the SPACE bar. To view one more line, press RETURN. Press **h** for help and **q** to quit.

optional The pipeline symbol (**|**) implies continuation. Thus the following command line

```
$ who | grep 'sam'
sam       tty1        2013-03-24 05:00
```

is the same as these command lines

```
$ who |
> grep 'sam'
sam       tty1        2013-03-24 05:00
```

When the shell parses a line that ends with a pipeline symbol, it requires more input before it can execute the command line. In an interactive environment, it issues a secondary prompt (**>**; page 368) as shown above. Within a shell script, it processes the next line as a continuation of the line that ends with the pipeline symbol. See page 1085 for information about control operators and implicit command-line continuation.

LPI Filters

A *filter* is a command that processes an input stream of data to produce an output stream of data. A command line that includes a filter uses a pipeline symbol to connect standard output of one command to standard input of the filter. Another pipeline symbol connects standard output of the filter to standard input of another command. Not all utilities can be used as filters.

In the following example, sort is a filter, taking standard input from standard output of who and using a pipeline symbol to redirect standard output to standard input of lpr. This command line sends the sorted output of who to the printer:

```
$ who | sort | lpr
```

The preceding example demonstrates the power of the shell combined with the versatility of Linux utilities. The three utilities who, sort, and lpr were not designed to work with one another, but they all use standard input and standard output in the conventional way. By using the shell to handle input and output, you can piece standard utilities together on the command line to achieve the results you want.

LPI tee The tee utility copies its standard input both to a file and to standard output. This utility is aptly named: It takes a single stream of input and sends the output in two directions. The next example sends the output of who via a pipeline to standard input of tee. The tee utility saves a copy of standard input in a file named **who.out** and also sends a copy to standard output. Standard output of tee goes via a pipeline to standard input of grep, which displays only those lines containing the string **sam**. Use tee with the **–a** (append) option to cause it to append to a file instead of overwriting it.

```
$ who | tee who.out | grep sam
sam        tty1         2013-03-24 05:00
$ cat who.out
sam        tty1         2013-03-24 05:00
max        pts/4        2013-03-24 12:23
max        pts/5        2013-03-24 12:33
zach       pts/7        2013-03-23 08:45
```

optional

LE+ Lists

A *list* is one or more pipelines (including simple commands), each separated from the next by one of the following control operators: **;**, **&**, **&&**, or **||**. The **&&** and **||** control operators have equal precedence; they are followed by **;** and **&**, which have equal precedence. The **;** control operator is covered on page 347 and **&** on page 348. See page 1085 for information about control operators and implicit command-line continuation.

An AND list has the following syntax:

pipeline1 && ***pipeline2***

where ***pipeline2*** is executed if and only if ***pipeline1*** returns a *true* (zero) exit status. In the following example, the first command in the list fails (and displays an error message) so the shell does not execute the second command (**cd /newdir**; because it is not executed, it does not display an error message):

```
$ mkdir /newdir && cd /newdir
mkdir: cannot create directory '/newdir': Permission denied
```

The exit status of AND and OR lists is the exit status of the last command in the list that is executed. The exit status of the preceding list is *false* because mkdir was the last command executed and it failed.

An OR list has the following syntax:

pipeline1 || ***pipeline2***

where ***pipeline2*** is executed if and only if ***pipeline1*** returns a *false* (nonzero) exit status. In the next example, the first command (ping tests the connection to a remote machine and sends standard output and standard error to **/dev/null**) in the list fails so the shell executes the second command (it displays a message). If the first command had completed successfully, the shell would not have executed the second command (and would not have displayed the message). The list returns an exit status of *true*.

```
$ ping -c1 station &>/dev/null || echo "station is down"
station is down
```

For more information refer to "&& and || Boolean Control Operators" on page 349.

LPI Running a Command in the Background

LPI Foreground All commands up to this point have been run in the foreground. When you run a command in the *foreground,* the shell waits for it to finish before displaying another prompt and allowing you to continue. When you run a command in the *background,* you do not have to wait for the command to finish before running another command.

Jobs A *job* is another name for a process running a pipeline (which can be a simple command). You can have only one foreground job in a window or on a screen, but you can have many background jobs. By running more than one job at a time, you are using one of Linux's features: multitasking. Running a command in the background can be useful when the command will run for a long time and does not need supervision. It leaves the screen free so you can use it for other work. Alternately, when you are using a GUI, you can open another window to run another job.

Job number, PID number To run a command in the background, type an ampersand (&; a control operator) just before the RETURN that ends the command line. The shell assigns a small number to the job and displays this *job number* between brackets. Following the job number, the shell displays the *process identification (PID) number*—a larger number assigned by the operating system. Each of these numbers identifies the command running in the background. The shell then displays another prompt, and you can enter another command. When the background job finishes, the shell displays a message giving both the job number and the command line used to run the command.

The following example runs in the background; it is a pipeline that sends the output of ls to lpr, which sends it to the printer.

```
$ ls -l | lpr &
[1] 22092
$
```

The **[1]** following the command line indicates that the shell has assigned job number 1 to this job. The **22092** is the PID number of the first command in the job. When this background job completes execution, you see the message

```
[1]+ Done          ls -l | lpr
```

(In place of **ls –l**, the shell might display something similar to **ls ––color=auto –l**. This difference is due to the fact that ls is aliased [page 398] to **ls ––color=auto**.)

LPI Moving a Job from the Foreground to the Background

CONTROL-Z and bg

You can suspend a foreground job (stop it from running) by pressing the suspend key, usually CONTROL-Z. The shell then stops the process and disconnects standard input from the keyboard. It does, however, still send standard output and standard error to the screen. You can put a suspended job in the background and restart it by using the bg command followed by the job number. You do not need to specify the job number when there is only one suspended job.

Redirect the output of a job you run in the background to keep it from interfering with whatever you are working on in the foreground (on the screen). Refer to "Control Operators: Separate and Group Commands" on page 347 for more detail about background tasks.

fg

Only the foreground job can take input from the keyboard. To connect the keyboard to a program running in the background, you must bring the program to the foreground. To do so, type **fg** without any arguments when only one job is in the background. When more than one job is in the background, type **fg**, or a percent sign (**%**), followed by the number of the job you want to bring to the foreground. The shell displays the command you used to start the job (**promptme** in the following example), and you can enter input the program requires to continue.

```
$ fg 1
promptme
```

LPI kill: Aborting a Background Job

The interrupt key (usually CONTROL-C) cannot abort a background process because the keyboard is not attached to the job; you must use kill (page 455) for this purpose. Follow **kill** on the command line with either the PID number of the process you want to abort or a percent sign (**%**) followed by the job number.

Determining the PID of a process using ps

If you forget a PID number, you can use the ps (process status) utility (page 380) to display it. The following example runs a find command in the background, uses ps to display the PID number of the process, and aborts the job using kill:

```
$ find / -name memo55 > mem.out &
[1] 18228
$ ps | grep find
18228 pts/10   00:00:01 find
$ kill 18228
[1]+  Terminated              find / -name memo55 > mem.out
$
```

LPI Determining the number of a job using jobs

If you forget a job number, you can use the jobs command to display a list of jobs that includes job numbers. The next example is similar to the previous one except it uses the job number instead of the PID number to identify the job to be killed. Sometimes the message saying the job is terminated does not appear until you press RETURN after the RETURN that executes the kill command.

```
$ find / -name memo55 > mem.out &
[1] 18236

$ bigjob &
[2] 18237

$ jobs
[1]-  Running                 find / -name memo55 > mem.out &
[2]+  Running                 bigjob &
$ kill %1
$ RETURN
[1]-  Terminated              find / -name memo55 > mem.out
$
```

LE+ Filename Generation/Pathname Expansion

Wildcards, globbing When you specify an abbreviated filename that contains *special characters,* also called *metacharacters,* the shell can generate filenames that match the names of existing files. These special characters are also referred to as *wildcards* because they act much as the jokers do in a deck of cards. When one of these characters appears in an argument on the command line, the shell expands that argument in sorted order into a list of filenames and passes the list to the program called by the command line. Filenames that contain these special characters are called *ambiguous file references* because they do not refer to one specific file. The process the shell performs on these filenames is called *pathname expansion* or *globbing*.

Ambiguous file references can quickly refer to a group of files with similar names, saving the effort of typing the names individually. They can also help find a file whose name you do not remember in its entirety. If no filename matches the ambiguous file reference, the shell generally passes the unexpanded reference—special characters and all—to the command. See "Brace Expansion" on page 411 for a technique that generates strings that do not necessarily match filenames.

The ? Special Character

The question mark (?) is a special character that causes the shell to generate filenames. It matches any single character in the name of an existing file. The following command uses this special character in an argument to the lpr utility:

```
$ lpr memo?
```

The shell expands the **memo?** argument and generates a list of files in the working directory that have names composed of **memo** followed by any single character. The shell then passes this list to lpr. The lpr utility never "knows" the shell generated the filenames it was called with. If no filename matches the ambiguous file reference, the shell passes the string itself (**memo?**) to lpr or, if it is set up to do so, passes a null string (see **nullglob** on page 408).

The following example uses ls first to display the names of all files in the working directory and then to display the filenames that **memo?** matches:

```
$ ls
mem   memo12  memo9  memomax   newmemo5
memo  memo5   memoa  memos

$ ls memo?
memo5  memo9  memoa  memos
```

The **memo?** ambiguous file reference does not match **mem**, **memo**, **memo12**, **memomax**, or **newmemo5**. You can also use a question mark in the middle of an ambiguous file reference:

```
$ ls
7may4report  may4report     mayqreport  may_report
may14report  may4report.79  mayreport   may.report

$ ls may?report
may4report  mayqreport  may_report  may.report
```

echo You can use echo and ls to practice generating filenames. The echo builtin displays the arguments the shell passes to it:

```
$ echo may?report
may4report mayqreport may_report may.report
```

The shell first expands the ambiguous file reference into a list of files in the working directory that match the string **may?report**. It then passes this list to echo, as though you had entered the list of filenames as arguments to echo. The echo utility displays the list of filenames.

A question mark does not match a leading period (one that indicates a hidden filename; page 188). When you want to match filenames that begin with a period, you must explicitly include the period in the ambiguous file reference.

The * Special Character

The asterisk (*) performs a function similar to that of the question mark but matches any number of characters, *including zero characters,* in a filename. The following example first shows all files in the working directory and then shows commands that display all the filenames that begin with the string **memo**, end with the string **mo**, and contain the string **alx**:

```
$ ls
amemo  memalx  memo.0612  memoalx.0620  memorandum  sallymemo
mem    memo    memoa      memoalx.keep  memosally   user.memo

$ echo memo*
memo memo.0612 memoa memoalx.0620 memoalx.keep memorandum memosally

$ echo *mo
amemo memo sallymemo user.memo

$ echo *alx*
memalx memoalx.0620 memoalx.keep
```

The ambiguous file reference **memo*** does not match **amemo**, **mem**, **sallymemo**, or **user.memo**. Like the question mark, an asterisk does *not* match a leading period in a filename.

The **–a** option causes ls to display hidden filenames (page 188). The command **echo *** does not display **.** (the working directory), **..** (the parent of the working directory), **.aaa**, or **.profile**. In contrast, the command **echo .*** displays only those four names:

```
$ ls
aaa  memo.0612  memo.sally  report  sally.0612  saturday  thurs

$ ls -a
.   aaa   memo.0612   .profile  sally.0612  thurs
..  .aaa  memo.sally  report    saturday

$ echo *
aaa memo.0612 memo.sally report sally.0612 saturday thurs

$ echo .*
. .. .aaa .profile
```

In the following example, **.p*** does not match **memo.0612**, **private**, **reminder**, or **report**. The **ls .*** command causes ls to list **.private** and **.profile** in addition to the contents of the **.** directory (the working directory) and the **..** directory (the parent of the working directory). When called with the same argument, echo displays the names of files (including directories) in the working directory that begin with a dot (**.**) but not the contents of directories.

```
$ ls -a
.  ..  memo.0612  private  .private  .profile  reminder  report

$ echo .p*
.private .profile

$ ls .*
.private .profile
.:
memo.0612  private    reminder   report
..:
...

$ echo .*
. .. .private .profile
```

You can plan to take advantage of ambiguous file references when you establish conventions for naming files. For example, when you end the names of all text files with **.txt**, you can reference that group of files with ***.txt**. The next command uses this convention to send all text files in the working directory to the printer. The ampersand causes lpr to run in the background.

```
$ lpr *.txt &
```

The shell expands ambiguous file references

tip *The shell does the expansion* when it processes an ambiguous file reference, not the program that the shell runs. In the examples in this section, *the utilities* (ls, cat, echo, lpr) *never see the ambiguous file references.* The shell expands the ambiguous file references and passes a list of ordinary filenames to the utility. In the previous examples, echo demonstrates this fact because it simply displays its arguments; it never displays the ambiguous file reference.

The [ ] Special Characters

A pair of brackets surrounding one or more characters causes the shell to match filenames containing the individual characters within the brackets. Whereas **memo?** matches **memo** followed by any character, **memo[17a]** is more restrictive: It matches only **memo1**, **memo7**, and **memoa**. The brackets define a *character class* that includes all the characters within the brackets. (GNU calls this a *character list;* a *GNU character class* is something different.) The shell expands an argument that includes a character-class definition by substituting each member of the character class, *one at a time,* in place of the brackets and their contents. The shell then passes the list of matching filenames to the program it is calling.

Each character-class definition can replace only a single character within a filename. The brackets and their contents are like a question mark that substitutes only the members of the character class.

The first of the following commands lists the names of all files in the working directory that begin with **a**, **e**, **i**, **o**, or **u**. The second command displays the contents of the files named **page2.txt**, **page4.txt**, **page6.txt**, and **page8.txt**.

```
$ echo [aeiou]*
...

$ less page[2468].txt
...
```

A hyphen within brackets defines a range of characters within a character-class definition. For example, **[6–9]** represents **[6789]**, **[a–z]** represents all lowercase letters in English, and **[a–zA–Z]** represents all letters, both uppercase and lowercase, in English.

The following command lines show three ways to print the files named **part0**, **part1**, **part2**, **part3**, and **part5**. Each of these command lines causes the shell to call lpr with five filenames:

```
$ lpr part0 part1 part2 part3 part5

$ lpr part[01235]

$ lpr part[0-35]
```

The first command line explicitly specifies the five filenames. The second and third command lines use ambiguous file references, incorporating character-class definitions. The shell expands the argument on the second command line to include all files that have names beginning with **part** and ending with any of the characters in the character class. The character class is explicitly defined as **0**, **1**, **2**, **3**, and **5**. The third command line also uses a character-class definition but defines the character class to be all characters in the range **0–3** plus **5**.

The following command line prints 39 files, **part0** through **part38**:

```
$ lpr part[0-9] part[12][0-9] part3[0-8]
```

The first of the following commands lists the files in the working directory whose names start with **a** through **m**. The second lists files whose names end with **x**, **y**, or **z**.

```
$ echo [a-m]*
...

$ echo *[x-z]
...
```

optional When an exclamation point (**!**) or a caret (**^**) immediately follows the opening bracket (**[**) that starts a character-class definition, the character class matches any character *not* between the brackets. Thus **[^tsq]*** matches any filename that does *not* begin with **t**, **s**, or **q**.

The following examples show that ***[^ab]** matches filenames that do not end with the letter **a** or **b** and that **[^b-d]*** matches filenames that do not begin with **b**, **c**, or **d**.

```
$ ls
aa  ab  ac  ad  ba  bb  bc  bd  cc  dd

$ ls *[^ab]
ac  ad  bc  bd  cc  dd

$ ls [^b-d]*
aa  ab  ac  ad
```

You can cause a character class to match a hyphen (**–**) or a closing bracket (**]**) by placing it immediately before the final (closing) bracket.

The next example demonstrates that the ls utility cannot interpret ambiguous file references. First ls is called with an argument of **?old**. The shell expands **?old** into a matching filename, **hold**, and passes that name to ls. The second command is the same as the first, except the **?** is quoted (by preceding it with a backslash [****]; refer to "Special Characters" on page 150). Because the **?** is quoted, the shell does not recognize it as a special character and passes it to ls. The ls utility generates an error message saying that it cannot find a file named **?old** (because there is no file named **?old**).

```
$ ls ?old
hold

$ ls \?old
ls: ?old: No such file or directory
```

Like most utilities and programs, ls cannot interpret ambiguous file references; that work is left to the shell.

Builtins

A *builtin* is a utility (also called a *command*) that is built into a shell. Each of the shells has its own set of builtins. When it runs a builtin, the shell does not fork a new process. Consequently builtins run more quickly and can affect the environment of the current shell. Because builtins are used in the same way as utilities, you will not typically be aware of whether a utility is built into the shell or is a stand-alone utility.

For example, echo is a shell builtin. It is also a stand-alone utility. The shell always executes a shell builtin before trying to find a command or utility with the same name. See page 1062 for an in-depth discussion of builtin commands and page 1077 for a list of bash builtins.

Listing bash builtins — To display a list of bash builtins, give the command **info bash shell builtin**. To display a page with information on each builtin, move the cursor to the **Bash Builtins** line and press RETURN. Alternately, you can view the **builtins** man page.

Getting help with bash builtins — You can use the bash **help** command to display information about bash builtins. See page 141 for more information.

Chapter Summary

The shell is the Linux command interpreter. It scans the command line for proper syntax, picking out the command name and arguments. The name of the command is argument zero. The first argument is argument one, the second is argument two, and so on. Many programs use options to modify the effects of a command. Most Linux utilities identify an option by its leading one or two hyphens.

When you give it a command, the shell tries to find an executable program with the same name as the command. When it does, the shell executes the program. When it does not, the shell tells you it cannot find or execute the program. If the command is a simple filename, the shell searches the directories listed in the **PATH** variable to locate the command.

When it executes a command, the shell assigns one file or device to the command's standard input and another file to its standard output. By default, the shell causes a command's standard input to come from the keyboard and its standard output to go to the screen. You can instruct the shell to redirect a command's standard input from or standard output to any file or device. You can also connect standard output of one command to standard input of another command to form a pipeline. A filter is a command that reads its standard input from standard output of one command and writes its standard output to standard input of another command.

When a command runs in the foreground, the shell waits for the command to finish before it displays a prompt and allows you to continue. When you put an ampersand (&) at the end of a command line, the shell executes the command in the background and displays another prompt immediately. Run slow commands in the background when you want to enter other commands at the shell prompt. The jobs builtin displays a list of suspended jobs and jobs running in the background and includes the job number of each.

The shell interprets special characters on a command line to generate filenames. A reference that uses special characters (wildcards) to abbreviate a list of one or more filenames is called an ambiguous file reference. A question mark represents any single character, and an asterisk represents zero or more characters. A single character might also be represented by a character class: a list of characters within brackets.

A builtin is a utility that is built into a shell. Each shell has its own set of builtins. When it runs a builtin, the shell does not fork a new process. Consequently builtins run more quickly and can affect the environment of the current shell.

Utilities and Builtins Introduced in This Chapter

Table 5-1 lists the utilities introduced in this chapter.

Table 5-1 New utilities

Utility	Function
tr	Maps one string of characters to another (page 167)
tee	Sends standard input to both a file and standard output (page 170)
bg	Moves a process to the background (page 172)
fg	Moves a process to the foreground (page 172)
jobs	Displays a list of suspended jobs and jobs running in the background (page 172)

Exercises

1. What does the shell ordinarily do while a command is executing? What should you do if you do not want to wait for a command to finish before running another command?
2. Using sort as a filter, rewrite the following sequence of commands:

```
$ sort list > temp
$ lpr temp
$ rm temp
```

3. What is a PID number? Why are these numbers useful when you run processes in the background? Which utility displays the PID numbers of the commands you are running?

4. Assume the following files are in the working directory:

```
$ ls
intro      notesb     ref2       section1   section3   section4b
notesa     ref1       ref3       section2   section4a  sentrev
```

   Give commands for each of the following, using wildcards to express filenames with as few characters as possible.

   a. List all files that begin with **section**.

   b. List the **section1**, **section2**, and **section3** files only.

   c. List the **intro** file only.

   d. List the **section1**, **section3**, **ref1**, and **ref3** files.

5. Refer to the info or man pages to determine which command will

   a. Display the number of lines in its standard input that contain the *word* **a** or **A**.

   b. Display only the names of the files in the working directory that contain the pattern **$(**.

   c. List the files in the working directory in reverse alphabetical order.

   d. Send a list of files in the working directory to the printer, sorted by size.

6. Give a command to

   a. Redirect standard output from a sort command to a file named **phone_list**. Assume the input file is named **numbers**.

   b. Translate all occurrences of the characters [and { to the character (, and all occurrences of the characters] and } to the character), in the file **permdemos.c**. (*Hint:* Refer to the tr man page.)

   c. Create a file named **book** that contains the contents of two other files: **part1** and **part2**.

7. The lpr and sort utilities accept input either from a file named on the command line or from standard input.

   a. Name two other utilities that function in a similar manner.

   b. Name a utility that accepts its input only from standard input.

8. Give an example of a command that uses grep

   a. With both input and output redirected.

   b. With only input redirected.

   c. With only output redirected.

   d. Within a pipeline.

   In which of the preceding cases is grep used as a filter?

9. Explain the following error message. Which filenames would a subsequent ls command display?

```
$ ls
abc  abd  abe  abf  abg  abh
$ rm abc ab*
rm: cannot remove 'abc': No such file or directory
```

Advanced Exercises

10. When you use the redirect output symbol (>) on a command line, the shell creates the output file immediately, before the command is executed. Demonstrate that this is true.

11. In experimenting with variables, Max accidentally deletes his **PATH** variable. He decides he does not need the **PATH** variable. Discuss some of the problems he could soon encounter and explain the reasons for these problems. How could he *easily* return **PATH** to its original value?

12. Assume permissions on a file allow you to write to the file but not to delete it.

    a. Give a command to empty the file without invoking an editor.

    b. Explain how you might have permission to modify a file that you cannot delete.

13. If you accidentally create a filename that contains a nonprinting character, such as a CONTROL character, how can you remove the file?

14. Why does the **noclobber** variable *not* protect you from overwriting an existing file with cp or mv?

15. Why do command names and filenames usually not have embedded SPACEs? How would you create a filename containing a SPACE? How would you remove it? (This is a thought exercise, not recommended practice. If you want to experiment, create a file and work in a directory that contains only your experimental file.)

16. Create a file named **answer** and give the following command:

    ```
    $ > answers.0102 < answer cat
    ```

    Explain what the command does and why. What is a more conventional way of expressing this command?

This page intentionally left blank

Main Index

An italic page number such as *123* indicates a definition. A light page number such as 456 indicates a brief mention. Page numbers followed by the letter **t** refer to tables. Only variables that must always appear with a leading dollar sign are indexed with a leading dollar sign. Other variables are indexed without a leading dollar sign.

Symbols

–– argument 155, 1011
^ in regular expressions 1164
^ quick substitution character 390
, (comma) operator 1083
; control operator 170, 347, 1085
;; control operator 1028, 1085
: (null) builtin 1059, 1071
:– substitutes default values for a variable 1059
::1 (IP address) 485
:? sends to standard error an error message for a variable 1060
:= assigns default values for a variable 1059
! (NOT) Boolean operator 1085
! event reference 387
!! reexecutes the previous event 388
? in extended regular expressions 1168
? special character 173
. (dot) builtin 338, 1067
. directory 194, 493
. in regular expressions 1163
./ executes a file in the working directory 344, 366
.. directory 194, 493
.jpg filename extension *1255*
‘ …‘ *see* command, substitution
( ) control operator 350
((…)) *see* arithmetic, evaluation
[] character class (regular expressions) 1163, *1238*
[] special characters 176
[…] *see* test utility
[[…]] builtin 1080
{} around positional parameters 1045
{} around variable names 361
{} expansion 411, 1018, 1073
{} in array variables 1060
{} in functions 402
@ (origin, DNS) 910
@ in a network address 307
* in regular expressions 1164
* special character 174
/ (root) directory *40*, 42, *185*, 189, 197, 617, *1270*
/ trailing within pathnames 40
/ within pathnames 40
\ escape character 150, 351, 360
\(in regular expressions 1166
\) in regular expressions 1166
& (AND) bitwise operator 1084
& control operator 170, 171, 348, 381, 1085
& in replacement strings (regular expressions) 1167
&& (AND) Boolean operator 535, 1079, 1084
&& control operator 170, 349, 1043, 1085
&> redirects standard output and standard error 171, 341
# comment 346, 1014
# prompt 597
#! specifies a script shell 344, 1014
+ in extended regular expressions 1168
< redirects standard input 163–164
<< Here document 1036–1038
> redirects standard output 162–163
>& duplicates output file descriptor 341, 1008
>&2 duplicates file descriptor 341, 1008
>> redirects and appends standard output 165
>| redirects output without **clobber** 165
| (OR) bitwise operator 1084
| (OR) Boolean operator (extended regular expression) 1168
| control operator 167, 348, 1021, 1083, 1085
| *see* pipeline
|& control operator 167, 341, 1085
|& shorthand for **2>&1** 341
|| (OR) Boolean operator 1079, 1084
|| control operator 170, 349, 1085
~– synonym for **OLDPWD** 414

~ (tilde) expansion 190, 365, 413
~ in directory stack manipulation 414
~ *see* home directory
~+ synonym for **PWD** 414
$ bash parameters 1044–1052
$ in regular expressions 1164
$ in variable names 359
$– parameter 1052
$_ parameter 1052
$! parameter 1051
$? parameter 1051
$(...) *see* command, substitution
$((...)) *see* arithmetic, expansion
${...} string length 1081
${} expands variables 1058
$@ parameter 1019, 1021, 1048
$* parameter 1048
$# parameter 1049
$$ parameter 1034, 1050
$0 parameter 1044
$*n* parameters 1045

Numerics

0< redirects standard input 340
000-default.conf file *955*
1> redirects standard output 340
1>&2 duplicates file descriptor 341, 1008
10.0.0.0 (IP address) 1232
100BaseT cable 291
10Base2 cable 291
10BaseT cable 291
1000BaseT cable 291
127.0.0.1 (IP address) 306, 485
127.0.1.1 (IP address) 485
172.16.0.0 (IP address) 1232
192.168.0.0 (IP address) 1232
2> redirects standard error 340
3-DES encryption 629
32-bit versus 64-bit Ubuntu 31
64-bit PC processor architecture 33
64-bit versus 32-bit Ubuntu 31
802.11 wireless specification *1232*

A

–a (AND) Boolean operator 1012, 1079
a2dismod utility *957*
a2dissite utility *958*
a2enmod utility *957*
a2ensite utility *958*
aborting execution *see* kill builtin
absolute pathnames 157, *189*, *1232*
access *1232*
Access Control Lists *see* ACLs
access permissions 199–211, *1232*
  change using chmod 201–203
  directory 205–206
  display using ls 200
  execute 343–344
  Nautilus 111
  setgid *see* setgid
  setuid *see* setuid
access.conf file *599*
ACLs 206–211, *1232*
  access rules 207–211
  default rules 210
  effective rights mask 208
  getfacl utility 207–211
  setfacl utility 207–211
acpi boot parameter *76*
acpid daemon 320
active window *1232*
active workspace *105*
ad hoc mode, wireless 664
addbanner shell script 1072
addition operators 1082
address *see the type of address you are looking for* (e.g., MAC address, IP address) or *see the specific address* (e.g., 192.168.0.0)
address mask *see* network, mask
address space, private *see* network, private address space
adfs filesystem 497
adm group *596*
admin group *596*
administrator 98, 426, *596*
  adm group *596*
  lpadmin group *596*
  sudo group *596*, 604, 609
  wheel group *625*
AES (Advanced Encryption Standard) 629
affs filesystem 497
AIDE utility 614
algorithm *1087*
alias 398–401, *1232*
  examples 400–401
  mail 794
  quotation marks in 399
  recursion 399
  recursive plunge 401
  substitution 410
alias builtin 398
alias.conf file *959*
aliases file 589, 794
aliases.db file 795
Alice (placeholder name) 627
Almquist Shell *see* dash shell
alphanumeric character *1232*
alternatives directory 483
amanda utility *569*
ambiguous file references *173*, *1232*
AMD64 processor architecture 33
anacron daemon 320
anacron file *575*
anacron init script 573
anacron utility *575*
anacrontab file 320, *575*
anacrontab utility 320
AND (**–a**) Boolean operator 1012, 1079
AND (**&**) bitwise operator 1084
AND (**&&**) Boolean operator 535, 1079, 1084
Andreessen, Marc 325
angle bracket *1232*
animate *1232*
anonymous FTP 759
ANSI 11
ansi terminal name 1175
anti-aliasing *1233*
Apache 952–998
  see also Apache containers; Apache directives
  000-default.conf file *955*

a2dismod utility 957
a2dissite utility 958
a2enmod utility 957
a2ensite utility 958
alias.conf file 959
apache2 daemon 954
apache2 directory 960
apache2 file 953, 959
apache2 init script 954
apache2.conf file 958, 959, 960, 962
apache2ctl file 959
apache2ctl utility 955, 990
authentication modules 994
CGI (Common Gateway Interface) 992, 995
.conf filename extension 960
conf-available directory 960
conf-enabled directory 960
configuration directives *see* Apache directives
configuring (Cacti) 677
containers *see* Apache containers
content negotiation *985*
content, adding 957
contexts *967*
directives *see* Apache directives
directory context 967
directory listings 986
document root *953*, 961
documentation 952
DSOs (dynamic shared objects) 952, 991
envvars file 960
error codes 997
filename extensions 966
filesystem layout 959–961
.htaccess context 967
.htaccess file 961, 967, 982, 994
html directory 957, 961
.htpasswd file 994
htpasswd file 959
httpd daemon 953
HTTPS protocol 992
https:// URI prefix 992
indexing 986
JumpStart: getting Apache up and running 955
LANG variable 953
locale 953
logresolve utility 973
logs 960
magic file 960
mod_perl.so file 959
modifying content 953
mods-available directory 957, 959
mods-enabled directory 957
modules 952, 957, 991
modules directory 959
more information 952
MPMs (multiprocessing modules) 996
MultiViews option 986
Perl code 995
PHP code 995
ports.conf file 960
prerequisites 954
privileged port 953
process *953*
public_html directory 965
Python code 995
redirects 984
reverse name resolution 973
role alias 964
root privileges 953
rotatelogs file 959
scripting modules 995
self-signed certificate 993–994
server *953*
server config context 967
sites-available directory 955, 958, 960
sites-enabled directory 958, 960
slash, trailing and redirects 984
Software Foundation 952
SSL 992–994
telnet utility 991
terminology 953
testing 956
threads 996
troubleshooting 990
type maps 985
user content, publishing 965
.var filename extension 985
virtual host context 967
virtual hosts 958, *986*, 986–990
webalizer utility 997
www-data group 953
Apache containers 967–971
<Directory> 967, 968
<Files> 968
<IfModule> 969
<Limit> 969
<LimitExcept> 970
<Location> 970
<LocationMatch> 971
<VirtualHost> 971
Apache directives *961*, 961–984
AddHandler 974, 985, 992
Alias 975
Allow 981
AllowOverride 982
Deny 982
DirectoryIndex 966
DocumentRoot 964
ErrorDocument 975
ErrorLog 974
Group 979
HostnameLookups 973
Include 958, 979
IncludeOptional 979
IndexOptions 976
Listen 962
LoadModule 958, 979
LogLevel 974
MaxClients *see* Apache directives, MaxRequestWorkers
MaxConnectionsPerChild 971
MaxRequestsPerChild *see* Apache directives, MaxConnectionsPerChild
MaxRequestWorkers 971
MaxSpareServers 972
MinSpareServers 972
Options 980
Order 983
Redirect 962
RedirectMatch 963
Require 984
ScriptAlias 981
security 981
ServerAdmin 956, 963
ServerName 956, 964
ServerRoot 959, 978

Apache directives, *continued*
ServerSignature 956, 978
ServerTokens 978
special 967–971
StartServers 972
Timeout 973
UseCanonicalName 973, 984
User 959, 981
UserDir 965
apache2 daemon 954, *see also* Apache
apache2 directory 960
apache2 file 953, 959
apache2 init script 954
apache2.conf file 958, 959, 960, 962
apache2ctl file 959
apache2ctl utility 955, 990
API *1233*
apic boot parameter 76
apm boot parameter 76
apmd daemon 320
append *1233*
append standard output using **>>** 165
applet *1233*
Application Switcher 102
application, textual 106
applications, desktop 1173t
applications.d directory 925
apropos utility 70, 137, 432
APT 511, 515
see also apt-get
apt cron script 518
apt file 519
apt.conf file 518
apt.conf.d directory 518
apt-cache utility 522
apt-file utility 515
apt-get 512–514, 519–522
cache 518
configuration files 518
dependencies *see* software packages, dependencies
local package indexes 518
software-properties-gtk utility 518
source code, download using apt-get 523
sources.list file 516
sources.list.d directory 516
update-notifier 518
apt cron script 518
apt file 519
apt.conf file 518
apt.conf.d directory 518
apt-cache utility 522
apt-file utility 515
apt-get 512–514, 519–522
see also APT
commands, list of 520
dependencies *see* software packages, dependencies
dist-upgrade command 522
install error 516
installing packages 512
JumpStart: installing and removing software packages using apt-get 512
log file, **apt** 519
options 520
purging configuration files 513
removing packages 513
update command 521
updating the package index 521
upgrade command 521
architecture, processor 32
archive *1233*
files 257–260
shell file 1037
tar utility 257–260
archives directory (APT) 518, 524
arguments *153*, *1233*
command line 1048
convert standard input into using xargs 268
testing 1007, 1015
arithmetic
evaluation (bash) 414, 1022, 1062, 1078–1079
expansion (bash) 414–416, 1083
expression *1233*
operators 1081t, 1081–1085
armor (ASCII) 645
ARP (Address Resolution Protocol) 303
arp utility 303
array *1233*
ASCII *1233*
ASCII armor 645
ascii file 1233
ASCII terminal *1233*
ash *see* dash shell
ASLR (address space layout randomization) 31
ASP (application service provider) *1234*
aspell utility 1023
assembly language 10
assignment operators 1083
asterisk special character 174
asymmetric encryption *see* encryption, public key
asynchronous communication *495*
asynchronous event *1234*
at utility 320, 484, 576
at.allow file 484
at.deny file 484
AT&T Bell Laboratories 3, 334
atd daemon 320
Athena, Project 15
ATM link 289
attachments *1234*
attribute, LDAP 831
auth.log file 492, 622, 1172
authentication *1234*
see also cryptography, authentication
Apache 994
OpenSSH 631, 714, 717
stack, PAM *621*
authorized_keys file 715, 730
auto.master file 864
autoconfiguration, IPv6 301
autofs directory hierarchy 864
autofs file 865
automatic mounting *1234*
automount 863–866
auto.master file 864
autofs file 865
home directory 863
available file (**dpkg**) 524, 526
avoided *1234*
awk utility *see* gawk utility
AWS 704–708
Axmark, David 1136
Azure, Microsoft 703

B

B language 10
back door *1234*
back ticks *see* command, substitution
background
  command, running in the 171
  desktop 115
  foreground versus 171
  jobs 171–173
  process *381*, *1235*
backports software package category 516
backslash escape character 150, 351, 360
backslash in replacement strings 1167
BACKSPACE key (erase character) 130
backup *568–573*
  amanda utility *569*
  cp utility 232
  cpio utility *571*, 572
  dump/restore utilities 572
  full *568*
  incremental *568*
  media *569*
  partition planning and 44
  rsync utility 749
  simple *572*
  tar utility 257–260, *569*, 571t, 572
  utilities *569*
base operators 1084
basename *189*, *1235*
basename utility 1033
bash 334, *1236*
  see also alias; bash history; bash variables; builtin; command; command line; operators; shell scripts
  alias *see* alias
  archive 1037
  arguments 1048
  arithmetic evaluation 414, 1022, 1062, 1078–1079
  arithmetic operators 1081t, 1081–1085
  arrays *see* bash variables, array
  background process 381
  builtins *see* builtins
  calling program, name of 1044
  command line *see* command line
  command not found error message 158, 344
  command substitution *see* command, substitution
  commands *see* command
  conditional expressions 1079
  control operator *see* control operator
  control structures *see* control structures
  debugging prompt 369, 1017
  directory stack 355–357
  expressions 1078–1085
  features 405–409
  file descriptors *1038*, 1038–1041
  functions 402–404, 1040, 1061–1062
  globbing *418*
  hash table 382
  help command 141
  history *see* bash history
  inheritance 1054
  logical evaluation 1079
  menu 1034
  No such file or directory error message 365
  operators *see* operators
  option flags, displaying 1052
  options, command line 405
  pathname expansion 360
  Permission denied error message 158, 343, 365
  prompt (**PS1**) 367
  prompt (**PS2**) 368
  prompt (**PS3**) 369, 1035
  prompt (**PS4**) 369, 1017
  quiz shell script 1094
  quotation mark removal 419
  recursion 1087
  redirection operators 342t
  sequence expression 412, 1018
  set, turns features on and off 406
  shopt, turns features on and off 406
  special characters 150, 372, 372t
  standard error *see* standard error
  standard input *see* standard input
  standard output *see* standard output
  startup files 335–339
  step values 1018
  string operators 1080t
  variables *see* bash variables
  word splitting *369*
  –x option 1016, 1088
bash history *382*, 382–398
  bind builtin 397
  C Shell mechanism, classic 387–392
  commands
    editing 386–387, 392–398
    reexecuting 384–392
    viewing 385–386
  event *382*
    designators 388t
    modifiers 392t
    numbers *383*, 388
    reference using **!** 387–392
  expansion 410
  history builtin 383
  INPUTRC variable 396
  quick substitution 391
  Readline Library 392–398
  Readline variables 396t
  substitute modifier 390
  variables 383, 383t
  word designators 389, 389t
bash parameters *358*, 358–371
  see also bash variables
  $– 1052
  $_ 1052
  $! 1051
  $? 1051
  $@ 1019, 1021, 1048
  $* 1048
  $# 1049
  $$ 1034, 1050
  $0 1044

bash parameters, *continued*
$*n* 1045
parameter null or not set error message 1060
positional *1044*, 1044–1049
special 1049–1052
substitution 359
bash variables *358*, 358–371, 1053–1062
see also bash parameters
@ subscript 1060
***** subscript 1060
array 1060
assigning values to 358
attributes 362–364
attributes, listing 363
BASH_ENV 337
braces around 361
CDPATH 370
COLUMNS 1035
completion *395*
default values, assigning 1059
default values, substituting 1059
DEFAULT_RUNLEVEL 435, 439
DISPLAY 474
EDITOR 386, 607
ENV 337
environment 1054
environment, display using printenv 1056
error messages, sending to standard error 1060
expansion 414
exported *see* bash variables, environment
global *see* bash variables, environment
HISTFILE 383
HISTFILESIZE 383
history 383, 383t
HISTSIZE 383
HOME 365
IFS 369–370
INPUTRC 396
keyword *359*, 364–371, 371t
LANG 374, *953*, 1176
LC_ALL 374
LC_COLLATE 374
LC_CTYPE 374
LC_MESSAGES 374
LC_NUMERIC 374
LINES 1035
local 1054
MAIL 366
MAILCHECK 367
MAILPATH 367
naming 358
noclobber 164–166
null, expanding 1058
OLDPWD 414
OPTARG 1074
OPTIND 1074
parameter substitution 359
PATH 158, 337, 365–366, 613, 614, 1033
PREVLEVEL 433
PRINTER 244
PS1 367, 368t
PS2 351, 368
PS3 369, 1035
PS4 369, 1017
PWD 414
quoting 360
RANDOM 1019, 1093
Readline 396t
readonly 362
removing 362
REPLY 1035, 1064
RESOLVCONF (DNS) 905
RUNLEVEL 433
shell 1053
SUDO_EDITOR 607
syntax 361
TERM 129, 1176
TZ 377
unexport 1056
unset using unset 362
unset, expanding 1058
user created 358, 359–362
VISUAL 607
BASH_ENV variable 337
.bash_history file 383
.bash_login file 336
.bash_logout file 336
.bash_profile file 336–338, 383, 403, 480
bash.bashrc file 484
.bashrc file 337–338, 480
bashrc file 337
baud *1235*
baud rate *1235*
Bazaar version control 511
BCPL language 10
BDB 831
beer, free 2
Bell Laboratories 3, 334
Berkeley DB 831
Berkeley Internet Name Domain *see* DNS
Berkeley UNIX 3, *1235*
Berners-Lee, Tim 325
best practices, least privilege *597*
bg builtin 172, 354
/bin directory 197
bin directory 198
bind builtin 397
bind directory 907, 912
BIND *see* DNS
bind9 file 905, 906
binding, key *1255*
BIOS 445, *1235*
DVD or flash drive, set to boot from 32
security 32
birthday shell script 1036
bit *1235*
bucket 166, *481*
depth *see* color depth
-mapped display *1236*
BitTorrent 531–533
BitTorrent, download Ubuntu using 48, 50
bitwise operators 1082, 1084
blank characters 360, *1236*, *1273*
blanks *150*
blkid utility 454, 502
block
device *496*, *1236*
devices, list using lsblk *665*
disk *1236*
number *1236*
special file *1236*
blocking factor *1236*
Blowfish encryption 629
.bmp filename extension 188

Bob (placeholder name) 627
Boolean operators *1236*
 ! (NOT) 1085
 && (AND) 535, 1079, 1084
 | (OR; extended regular expression) 1168
 || (OR) 1079, 1084
 –a (AND) 1012, 1079
 control *see* && control operator *and* || control operator
 –o (OR) 1079
boot *1236*
 failure 439
 flag 85
 loader *1236*
 netboot *1261*
 options 75–77
 parameter 76
  modifying 75–77
  quiet 58
  splash 58
 system, the *56*, 438
/boot directory 43, 197, 445, 581
Boot menus, installation 72–75
bootable flag 85
bootstrap *1236*
Bourne, Steve 334, 1236
Bourne Again Shell *see* bash
Bourne Shell (original) 334, *1236*
brace *1236*
brace expansion 411, 1073
braces, variables and 361
bracket *1236*
bracket, character class 1163
branch *1236*
break control structure 1027
bridge, network *292*, *1237*
broadcast *1237*
 address *1237*
 network 288, *1237*
browsers *325*, 326
 file *see* Nautilus
 Mosaic 325
BSD *see* Berkeley UNIX
btmp file 584
buffer *1237*
 copy *107*
 primary *107*
 selection *107*
bug *535*, *1237*
 BugSplat *536*
 Bugzilla *536*
 defect-tracking system *511*, *536*
 Launchpad 511
builtins *178*, 1062–1077, 1077t, *1237*
 : (null) 1059, 1071
 . (dot) 338, 1067
 [[...]] 1080
 alias 398
 bg 172, 354
 bind 397
 cd 193, 216, 370
 commands that are symbols 339t
 declare 363–364, 1060
 dirs 355
 echo 134, 174, 227, 1031, 1031t, 1050
 eval 404, 1073
 exec 1039, 1067–1069
 executing 381
 exit 128, 1008, 1051
 export 363, 1054–1056
 fc 384–387
 fg 172, 353
 getopts 1074–1076
 history 383
 jobs 131, 172, 352
 kill 131, 172, 455–457, 1070, 1072
 let 416, 1078
 list using info 178
 local 403, 1062
 : (null) 1059, 1071
 popd 357
 pushd 356
 pwd 151, 192, 193, 356, 416
 read 1025, 1063–1065, 1065t, 1065–1066
 readonly 362, 363
 set 406, 1013, 1016, 1046, 1058
 shift 1011, 1047
 shopt 406
 source 338
 symbols as commands 339t
 test 1005–1007, 1007t, 1008, 1012, 1015, 1018, 1022, 1027
 tput 1026
 trap 1026, 1069–1072
 type 1063
 typeset *see* builtins, declare
 umask 459
 unalias 399, 401
 unset 362
 utilities versus 1006
bundle shell script 1037
bunzip2 utility 256
busybox utility 78
button, gear 108
by-path file 481
byte *1237*
bytecode *1237*
by-uuid file 481
.bz2 filename extension 188
bzcat utility 256
bzgrep utility 256
bzip2 utility 253–257, 570, 572
bzless utility 256

C

.c filename extension 188
C locale 375
C programming language 9, 10, *1237*
C++ programming language 11
C89 programming language 11
CA (Certificate Authority) *1238*
cable modems *1238*
cables 291
cache *1238*
cache, DNS *see* DNS, cache; DNS servers, cache
Cacti 674–683
 cacti (crontab) file 678
 cacti.conf file 678
 configuring 678
 debian.php file 677
 firewall setup 676, 681
 remote data source 681
 SNMP 681
cacti (crontab) file 678
cacti.conf file 678

cancel utility *557*
Canonical 35
cascading stylesheet *see* CSS
cascading windows *1238*
case control structure 1028–1034
case-sensitive *1238*
cat utility 133, 160, 162, 164, 224, 491, 721, 1011
categories, software package 122, 516
category *n* cables 291
catenate *163*, *224*, *1238*
cd builtin 193, 216, 370
CD device 481
CDPATH variable 370
CD-ROM, mount 500
CERN 325
certificate authority *see* CA
certificate, SSL *see* SSL, certificate
certification, CompTIA 1183–1229
certification, LPI 1183–1229
.cgi filename extension 966
CGI scripts (Apache) 992
chage utility 489, *567*
chain loading *1238*
character
  alphanumeric *1232*
  -based *1238*
  -based interface *see* command line; textual, interface
  -based terminal *1238*
  blank *150*, 360, *1236*, *1273*
  class *176*, 1169t, *1238*
  control *1241*
  device *496*, *1239*
  escaping 150, 351
  list *see* character, class
  map *374*
  meta *1259, see also* special characters
  nonprinting *1263*
  printable *1266*
  quoting 150, 351
  regular *1269*
  special *see* special characters
  special file *1239*
  typeface conventions 27
Charlie (placeholder name) 627
charmap *374*
CHARSET *374*
check *see* tick
check box 26, *1239*
check mark *see* tick
checksum *1239*
chgrp utility 203
child directories 185, 186
child processes 380, *1239*
Chinese Ubuntu 38
chkargs shell script 1006, 1008
chmod utility 201–203, 343
chown utility 203
chroot jail 617–621
  BIND 917
  DNS 917
  FTP 769
  named daemon 917
chroot utility 617–621
chsh utility 335, 454
CIDR 306, *1239*
CIFS *871*, *1239*
CIPE *1239*
cipher *627*, *1239*
ciphertext *627*, *1240*
Clark, Jim 325
class, character *1238*
Classless Inter-Domain Routing *see* CIDR
clear utility 454
cleartext *627*, *1240*
CLI *1240, see also* command line; textual interface
click and right-click 105
client *1240*
client, specifying 461t
client/server model 316
clipboard 108
clock, system, display using date 226
clock, system, set using the Date & Time window 117
clock, system, set using timedatectl 579
cloud 703–708, *1240*
CMOS setup 32
CN, LDAP *832*, 832
coaxial cable 291
coda filesystem 497
Codd, E. F. 1137
code, reentrant *1268*
CODEC *1240*
collating sequence, machine *1258*
collision domain *292*
color depth *1240*
color quality *see* color depth
column *1137*
COLUMNS variable 1035
combo box *1240*
Comer, Doug 5
comma operator 1083
command *152*, *1240*
  see also builtins; command line
  arguments 153
  arguments, convert standard input into using xargs 268
  completion 394–395
  continuing 351
  control flow *see* control structures
  editing/repeating 131
  execute using exec 1067–1069
  executing 381
  execution environment *1054*
  grouping 350
  –h option 155
  ––help option 154
  human-readable option 155
  interpreter, shell *128*
  line *see* command line
  names 152
  network extension 307
  run remotely using ssh 720–722
  separating 347–351
  simple *152*, *155*
  substitution *416*, 416–417, 447, 1015, 1033, *1240*
command line 125–142, *152*, 152–159, *1240*
  see also command; shell
  –– argument 155, 1011
  advantages of 125
  arguments 153, 1045
  arguments, initialize using set 1046
  arguments, promote using shift 1047
  brace expansion 411, 1073
  continuation 169, 351, 1085

control operators 170
editing 131, *392–398*
executing 158
expansion 410–419
filters 169
interface *1240, see also* textual, interface
lists 170
mistakes, correcting 129
options *153*, 153–155, 405, 405t
parse 156, *410*
print utilities 557t
printing from the 557
processing 156–158, 409
syntax 152
tokens *153*, 410
whitespace on the 348
words *153*, 410
command not found error message 158, 344
command_menu shell script 1030
comments, MariaDB 1138
comments, shell scripts 346, 1014
Common Name, LDAP *832*
Common UNIX Printing System *see* CUPS
communication, asynchronous *495*
communication, interprocess *495*
comparison operators 1082
completion
command 394–395
filename *1247*
pathname 395
Readline *394*
Readline commands 394–395
variable 395
component architecture *1240*
compress utility 257, 260
compress/decompress files 253–257
bunzip2 utility 256
bzcat utility 256
bzgrep utility 256
bzip2 utility 253–257, 572
bzless utility 256
compress utility 257, 260
gunzip utility 256
gzip utility 253–257, 260
OpenSSH 738
unxz utility 256
unzip utility 257
utilities 281t
WinZip utility 257
xz utility 253–257
xzcat utility 256
xzgrep utility 256
xzless utility 256
zcat utility 256
zgrep utility 256
zip utility 257
CompTIA certification 1183–1229
computer, diskless *1244*
computing, distributed *1244*
concatenate *see* catenate
concentrator *see* hub
condition code *see* exit status
conditional evaluation operators 1083
conditional expressions 1079
.conf filename extension 461, 960
conf-available directory 960
conf-enabled directory 960
confidentiality *see* cryptography, confidentiality
config file (OpenSSH) 715, 726
configuration file rules 461
Configure and Build System, GNU *534*
configure shell script *534*
connectionless protocol *1241*
connection-oriented protocol *297*, *1241*
console *1241*
recovery mode 450
security *597*
virtual 61, *127*, *1280*
context menu *106*
continuation, command line 169, 1085
continue control structure 1027
control character *1241*
CONTROL characters, display using cat 225
control flow *see* control structures
CONTROL key *27*
control operator *347*, 347–352, 1241
; 170, 347, 1085
;; 1028, 1085
( ) 350
& 170, 348, 1085
&& 170, 349, 1043, 1085
| 167, 348, 1021, 1083, 1085
|& 167, 341, 1085
|| 170, 349, 1085
continuation, implicit command line 1085
NEWLINE 347
short-circuiting *349*
control structures 1004–1038, *1241*
break 1027
case 1028–1030, 1030t, 1030–1034
continue 1027
for 1019–1021
for...in 1017–1020
Here document 1036–1038
if...then 1005–1008
if...then...elif 1011–1016
if...then...else 1009–1011
Python 1114–1118
select 1034–1036
until 1025–1027
while 1021–1024
CONTROL-\ key (quit) 131
CONTROL-ALT-DEL 442
control-alt-delete.conf file 442
CONTROL-C key (copy) 107
CONTROL-C key (interrupt) 130
CONTROL-D key (EOF) 128, 161
CONTROL-H key (erase character) 130, 156
CONTROL-U key (line kill) 130, 156
CONTROL-V key (paste) 107
CONTROL-V key (quote CONTROL keys) 151
CONTROL-W key (erase word) 130, 156
CONTROL-X key (cut) 107
CONTROL-X key (line kill) 130
CONTROL-Z key (suspend) 130, 172
convention, end line key 27

conventions used in this book 26–28
cookie *1241*
Coordinated Universal Time *see* UTC
copy buffer *107*
copyleft *5*
core file 591
correcting typing mistakes 129
count shell script 1022
count_down function 1062
country code domain name designation 316
cp utility 133, 153, 196, 232
cp versus ln 213
cpdir shell script 351
cpio utility 571, 572
CPU *1241*
  architecture 32
  information about, display 491
  installation requirements 31
  intensive processes, report on using top 577
  virtualization extensions 692
cpuinfo virtual file 241
crack utility 615
cracker *1242*
crash *443*, *1242*
creation date of files, display using ls 200
cron daemon 320, 573
cron.allow file 484
cron.d directory 320, 573, 575
cron.deny file 484
crontab *573*
crontab file 320, 574
crontab files 573
crontab utility 320, 484
crontabs directory 320, 573
cryptography *626*, *1242*
  see also encryption; GPG; hash function; PGP
  Alice (placeholder name) 627
  asymmetric key 629
  authentication *626*, *629*, *637*
  Bob (placeholder name) 627
  Charlie (placeholder name) 627
  ciphertext *627*
  cleartext *627*
  confidentiality *626*, *628*, *629*, *637*
  data integrity *626*
  decryption *627*
  digital signature *627*, *629*
  fingerprint *627*
  hash function *see* hash function
  integrity versus authenticity 634
  key *627*, 628
  key space 628
  key, OpenSSH 631
  Mallory (placeholder name) 627
  message *627*
  nonrepudiation of origin *626*
  passphrase *627*
  plaintext *627*
  public key *629*
  salt 634
  signing *627*, *629*
  SSL certificate *see* SSL certificate
csh Shell *1237*
.cshrc file 1242
.csr filename extension 638
CSRG (Computer Systems Research Group) 3
CSS *1242*
CUPS *540*
  see also printer; printing
  command-line interface *551–555*
  configuring a local printer automatically 542
  configuring a printer *550–551*
  cups init script 541
  cupsd.conf file *555*
  cupsdisable utility *555*
  cupsenable utility *555*
  cupsys init script 541
  drivers, display using lpinfo 552
  firewall setup 542
  IPP protocol 540
  JumpStart: configuring a printer using system-config-printer 542
  JumpStart: setting up a local or remote printer 544
  more information 541
  ppd directory 553
  PPD files *552*
  prerequisites 541
  print queue, managing 544, *555*
  setting up a local or remote printer using the CUPS Web interface 548
  URIs 546, *552*
  Web interface 548–549, *550–551*
cups init script 541
cupsaccept utility *555*
cups-browsed utility 437
cupsd.conf file *555*
cupsdisable utility *555*
cupsenable utility *555*
cupsreject utility *555*
cupsys init script 541
curl utility *536*
current *1242*
current directory *see* working directory
cursor *1242*
cut and paste 107, 321
cut utility 233, 415
cycling, window *102*
cypher *1239*

D

daemons 320t, *1243*
  see also service
  in. prefix 319
  messages 492
  network *286*, 319
  rpc. prefix 319
  start and stop using sysv-rc-conf 436–437
daily file 575
dash shell 13, 335
Dash, the 101
data integrity *see* cryptography, data integrity
data link layer, IP model protocol *297*
data sink 166
data structure *1243*
database *1137*
  see also MariaDB
  Berkeley 831

dbm *1243*
gdbm *1243*
locate 70
makewhatis 137
mandb 70, 137
ndbm *1243*
NIS *1262*
printcap 487
Sleepycat 831
SQL *1274*
datagram, network *1243*
datagram-oriented protocol *298*
dataless system *846*, *1243*
date utility 134, 226, 376, 377, 604, 1046
date, set using the Date & Time window 117
date, set using timedatectl *579*
DB_CONFIG file 834
db.127 file (DNS) 914
db.local file (DNS) 913
db.root file (DNS) 912
dbm database *1243*
DC, LDAP *831*
dd utility 482, 483, 491, 722
DDoS attack *1243*
.deb filename extension 511, 524
deb files *524*
Debian Almquist Shell *see* dash shell
debian.php file (Cacti) 677
debian-installer 78–82
debug *1243*
bash prompt 369, 1017
FTP 763
NIS 820, 829
scp using **–v** 725
server using telnet 310
shell scripts using **–e** and **–u** 345
shell scripts using **–x** 1016
shell scripts using **xtrace** 1088
ssh using **–v** 723, 735
sshd using **–d** 732, 735
DEBUG signal 1070
declare builtin 363–364, 1060
decompress files *see* compress/decompress files
decrement operators 1083
default *1243*
default directory 484
DEFAULT_RUNLEVEL variable 435, 439
defaultdomain file (NIS) 818
defect-tracking system *511*, *536*
DEL key (erase character) 130
delete character using BACKSPACE 129
delete line using CONTROL-U 130
delete word using CONTROL-W 130
delimiter, regular expression 1162
delta, SCCS *1243*
demand mounting, filesystem *864*
denial of service *see* DoS attack; DDoS attack
dependencies *510, see also* software packages, dependencies
depmod utility 444
dereference *1243*
derivatives, Ubuntu 37
descriptors, file *340*
desktop 17, 104–108, *1244*
active workspace *105*
alternate 37
appearance 115
Application Switcher 102
applications 1173t
background 115
context menu *106*
Dash, the 101
environment, Unity 17
focus *1248*
GNOME Flashback 103–104
Guide, Ubuntu 118
install different 124
manager *17*, *1244*
object *105*
panel *105, see also* panel
Run a Command window 106
Search screen *101*
System Settings window 113
terminology 105
Unity 17, 100–103
unity-tweak-tool utility 102
window *see* window
workspace *105*
Desktop edition 36
Desktop Image 71
detached process *see* background, process
/dev directory 160, 197, 481–483, 494
devfs filesystem 494
device *1244*
block *496*, *1236*
character *496*, *1239*
drivers 494, 496, *1244*
filename *1244*
files *160*, *494*, *1244*
files, exporting 857
hotplug 495
independence 14
major number *496*, *1258*
MD 86
minor number *496*, *1260*
multidisk 86
names, dynamic (udev) 494
null 166, 481
number, major *496*, *1258*
number, minor *496*, *1260*
physical *1265*
pseudoterminal 482
raw *496*
raw mode *496*
special files *see* device, files; special files
terminal 1068
UUID numbers 481
devpts filesystem 497
df utility 846, 866
dhclient utility 465
dhclient-*-*interface*.lease file 465
DHCP 464–468, *1244*
dhclient utility 465
dhclient.conf file 465
dhclient-*-*interface*.lease file 465
dhcpd daemon 465, 466
dhcpd.conf file 466, 468
IPv6 302
isc-dhcp-server init script 466
MAC addresses 468
prerequisites, client 465
prerequisites, server 466
static IP addresses 467
dhcpd daemon 465, 466
dhcpd.conf file 466, 468

dialog box *1244*
dial-up connection 128
dictionaries, Python 1113–1114
die, process 381
diff utility 235
diff3 utility 235
Diffie-Hellman encryption 629
dig utility 314, 901, 902–903, 906, 919
digest, hash *632*
digital signature *627*, *629*
directory *12*, *151*, *185*, *493*, *1244*, *see also* the File Tree index (1287)
  . 194, 493
  .. 194, 493
  / (root) 189, 197
  ~ (home) *see* home directory
  access permissions 205–206
  access, speed up using tune2fs 505
  change using cd 193
  child 185, 186
  compacting 592
  create using mkdir 192–193
  current *see* working directory
  delete using rmdir 194
  file *151*, *185*, *1244*
  folder and *108*
  hierarchy *40*, *1244*
  home *see* home directory
  important 480
  LDAP *830*
  links to 211–217
  list using ls 132, 229
  make using mkdir 192–193
  mount remote using NFS 848–852
  move using mv 196
  moving (inodes) 493
  parent 185, 186
  pathname *185*
  remove using rmdir 194
  rename using mv 196
  root (/) *40*, 42, *185*, 189, 197, 617, *1270*
  service *1244*
  stack *355*, *355–357*, 414
  standard 197–199
  subdirectories *185*
  tree *184*, *see also* directory, hierarchy
  working *see* working directory
dirs builtin *355*
disk
  block *1236*
  encrypt using gnome-disks 90
  encryption 30
  filesystem *39*
  formatting, low-level 38
  fragmentation 590
  free space *38*, *591*, *1249*
  hot-swap *45*
  LBA addressing mode 445
  monitor using SMART 91
  partition *see* partition
  quotas 592
  RAM 60, *1268*
  space, installation requirements 32
  usage, display using du 501
  usage, monitoring 590
  utility, gnome-disks 88–91
  volume label 455
diskless system *845*, *1244*
display
  bit-mapped *1236*
  color depth *1240*
  configuring 116
  graphical *1249*
  number, X Window System 474
––display option, X Window System 475
DISPLAY variable 474
displaying
  see also displaying a file
  date using date 134, 226
  file classification using file 237
  file information using ls 229
  hidden filenames using ls 175, 229
  kernel messages using dmesg 439, 454
  machine name using hostname 227
  PID using pidof 457
  text using echo 134, 227
displaying a file
  beginning of, using head 243
  cat 133, 224
  classification using file 237
  end of, using tail 249
  group, using ls 200
  hidden using ls 175, 188, 229
  less 228
  links, number of using ls 200
  more 228
  number of lines/words/bytes in using wc 252
  owner of, using ls 200
  size of, using ls 200
  sorted, using sort 247–249
  type of, using ls 200
distributed computing *315*, *1244*
distribution, Linux *6*
division operator 1082
dmesg utility 439, 454
DMZ *1244*
DN, LDAP *831*
DNS 316–318, 892–904, *1245*
  see also DNS records; DNS servers; DNS zones
  @ (origin) 910
  address, look up using host 313
  authority 895
  BIND 318
  bind directory 907, 912
  bind9 file 906
  bind9 init script 904
  cache *897*, 901
  cache, setting up 911–914
  chroot jail 917
  configuring 907–910
  database 897
  db.127 file 914
  db.local file 913
  db.root file 912
  delegation 895
  dig utility 314, 901, 902–903, 906, 919
  dnssec-keygen utility 915
  domain *893*, *894*
    name *893*
    qualification 910
    server *892*
  error messages 905

firewall setup 905
FQDN *894*
hints zone 912
host utility 313, 900, 903
in-addr.arpa domain 902
inverse mapping *see* DNS, reverse name resolution
ip6.int domain 902
iterative queries 896
JumpStart: setting up a DNS cache 906
log 919
more information 904
name, look up using host 313
named daemon 905
named directory 917
named.conf file 907, 911–912
named.conf.options file 912
nameserver *892*
node 893
nsswitch.conf file 903
origin *900*
origin *see* DNS zones, name
prerequisites 904
queries 896, 901
recursive queries 896
resolv.conf file 905, 906
resolvconf utility 488, 905
RESOLVCONF variable 905
resolver 894
reverse mapping *see* DNS, reverse name resolution
reverse name resolution 902–903
root domain *893*, 893, 895
root node *893*
security 893
server *892*
subdomain 894
terminology 905
time format 910
troubleshooting 919
TSIGs (transaction signatures) 915–917, 920
TTL value 900
working directory is not writable error message 905
DNS records
A (address, IPv4) 898
AAAA (address, IPv6) 302, 898
CNAME 898
glue 914
MX 899
NS 899
PTR 899
resource 898–900
SOA 899
TXT 900
DNS servers *892*
cache *897*, 906
caching only *897*
primary master *896*
secondary *897*
slave *897*
types of 896
DNS zones 895
clause, **named.conf** 909
files 910, 912
hint file 912
name 910
root 912
dnssec-keygen utility 915
doc directory 141, 199, 1172
Document Object Model *see* DOM
documentation *see* help
DOM *1245*
domain
see also DNS
DNS *893*, *894*
in-addr.arpa 902
ip6.int 902
name *893*, *1245*
country code 316
not case-sensitive 317
NIS 814
root 895
server *892*
VM *688*
Domain Name Service *see* DNS
door *1245*
DoS attack *1245*
DOS files, convert from/to Linux format using unix2dos/dos2unix 268
dos2unix utility 268
double quotation marks *see* quotation marks
double-click timeout, mouse 116
Dovecot IMAP and POP servers 807
dovecot, firewall setup 808
dovecot, starting 808
downloading Ubuntu 29, 47–50
dpkg *510*, 524–530
deb file, contents 524
deb files *524*
dpkg utility 526–530
dpkg utility letter codes 527t
postinst script 525
preinst script 525
prerm script 525
source files 525
dpkg utility 526–530
dpkg-reconfigure utility 789, 796
DPMS *1245*
drag *1245*
drag-and-drop *1245*
drive, optical *1263*
drivers, device 494, *1244*
drop-down list *1245*
druid *1246*
DSA (Digital Signature Algorithm) *1246*
DSA, LDAP *831*
DSE, LDAP *831*
DSL 289, *1246*
DSO, Apache 952
du utility 501
dual-boot system 91–93
dump utility 503, 572
duplex network *291*
DVD device 481
DVD, Ubuntu *see* installation image
Dynamic Host Configuration Protocol *see* DHCP
dynamic IP address 304, 670
dynamic shared objects, Apache 952

E

e2label utility 455
echo builtin 134, 174, 227, 1031, 1031t, 1050
echo utility 1050

ed utility 240
Edbuntu 37
edition *see* Desktop Image; Server Image; Netboot Image
EDITOR variable 386, 607
editors *1246*
  command line 392–398
  ed 240
  EDITOR variable 607
  gnome-disks 88–91
  nano 277–280
  parted *579–582*
  Readline Library 392–398
  root privileges, running with 607
  SUDO_EDITOR variable 607
  vi *see* vim
  vim *see* vim
  VISUAL variable 607
edquota utility *593*
educational derivative of Ubuntu 37
Edwards, Dan 1278
EEPROM *1246*
effective UID *599*, *1246*
egrep utility 241, 692, 1168
element *1246*
ElGamal encryption 629
email *see* mail; **postfix**
emoticon *1246*
empty regular expressions 1166
emulator, operating system *8*
emulator, terminal *126*, 128
encapsulation *see* tunneling
encrypt
  see also encryption
  disk using gnome-disks 90
  home directory 67
  partition 90
  Ubuntu installation 64
encryption *627*, 627–632, *1246*
  see also cryptography; encrypt; GPG; hash function
  3-DES 629
  AES 629
  algorithm *627*
  asymmetric *see* encryption, public key
  Blowfish 629
  Diffie-Hellman 629
  disk 30
  ElGamal 629
  hybrid 630
  IDEA 629
  OpenSSH 631, 714
  PGP 641
  private key 629
  public key 629, *629*
  RC5 629
  RSA 629, *1271*
  secret key *see* encryption, symmetric key
  symmetric key 628, *628*
  web of trust 641
end line key 27
end of file *see* EOF
Enquire program 325
ENTER key 27
entropy 642, *1246*
entropy, lack of hangs system 482, 642
entry, LDAP *831*
env utility 404, 1057
ENV variable 337
environment file 716
environment variable *see* variables, environment
environment, command execution *1054*
envvars file 960
EOF *1246*
EOF signal, send using CONTROL-D 161
epositories 1182
EPROM *1246*
–eq relational operator 1079
equality operators 1082
erase key (CONTROL-H) *129*, 156
erase word key (CONTROL-W) 156
erasing a file completely 482
ERR signal 1070
error messages
  see also messages; usage messages
  404 Not Found (Apache) 997
  Apache 997
  command not found 158, 344
  DNS 905
  Login incorrect 440
  mount: RPC: Program not registered 862
  NFS 862
  NFS server *xxx* not responding 862
  No such file or directory 365
  parameter null or not set 1060
  Permission denied 158, 343, 365
  redirecting to standard error 341, 1008
  rlimit_max (testparm) 887
  send to standard error for a variable using **:?** 1060
  server not responding (NFS) 851
  Stale NFS file handle 862
  standard error *see* standard error
  system 492
error, standard *see* standard error
errors, correcting typing mistakes 129
errors, lack of entropy hangs system 482, 642
escape a character *150, see also* quotation marks; quoting
ESXi 690
etab file 861
/etc directory 198, 483, 483–490
Ethernet address *see* MAC address
Ethernet network 290, *1247*
Eucalyptus 703
eval builtin 404, 1073
event 432, *1247*
  asynchronous *1234*
  bash history *see* bash history
  Upstart *428*
  X Window System *471*
evince utility 188
Evolution 800
exabyte *1247*
exception, Python *1120*
exec builtin 1039, 1067–1069
exec() system call 380
execute
  access permission *199*, 343–344

commands 158, 381
files in the working directory 344
shell scripts 346
exim4 783
exit builtin 128, 1008, 1051
EXIT signal 1069
exit status 416, 1006, 1008, 1051, *1247*
expansion
arithmetic (bash) 414–416, 1083
command line 410–419
pathname *173*
tilde 190
exploit *1247*
exponentiation operator 1082
export builtin 363, 1054–1056
export, device files 857
export, links 857
exported variable *see* environment variable
exportfs utility 856, 861
exports file 855, 857–860, 861
expressions *1247*
arithmetic *1233*
logical *1257*
regular *see* regular expression
ext2/ext3/ext4 filesystem 497, 504
extended regular expressions *see* regular expression, extended
Extensible Markup Language *see* XML
extensions, filename *see* filename, extensions
extranet *287*, *1247*
extras software package category 516

F

Fahlman, Scott 1273
failsafe session *1247*
fake RAID 45
false utility 349, 487
Favorites list 101
fc builtin 384–387
FDDI *1247*
fg builtin 172, 353
FHS (Linux Filesystem Hierarchy Standard) 12, 197
Fiber Distributed Data Interface *see* FDDI
FIFO special file *495*
fifth layer, IP model protocol *297*
file *12*, *1247*
see also displaying a file; filename
access permissions *see* access permissions
access time, change using touch 251
ambiguous references *173*
archiving 257–260
backup *see* backup
block special *1236*
browser *see* Nautilus
character special *1239*
compare using diff 235
compress *see* compress/decompress files
configuration, rules 461
contents, identify using file 237
convert from/to Linux/Windows format using unix2dos/dos2unix 268
copy over a network using rsync 748
copy using cp 133, 195, 232
create empty using touch 134, 251
create using cat 162
creation date, display using ls 200
creation mask, specify using umask 459
crontab 573
deb 524
decompress *see* compress/decompress files
descriptors *340*, *1038*, 1038–1041, 1042
device *see* **/dev** directory; device files; special files
directory *see* directory
display *see* displaying a file
edit using nano 277–280
edit using vim 270–277
erase completely 482
file utility 237, 492, 592
find using find 237–240
group assignment 485
group, change using chgrp 203
group, display using ls 200
growing quickly 591
handle, Python *1119*
hidden *188*, *1251*
hidden, display using ls 175, 188, 229
important 480
information about, displaying using ls 229
inode *see* inodes
invisible *see* filename, hidden
ISO image 47
job definition (Upstart) 429
links to 211–217
links, display number of using ls 200
log, checking 590
manager *see* Nautilus
map 865
modification time, change using touch 251
move using mv 195, 245
moving (inodes) 493
names *see* filename
open using Nautilus 110
open, locate using lsof 589
order using sort 247–249
ordinary *151*, *185*, *493*, *1263*
owner, change using chown 203
owner, display using ls 200
pathname *185*
permissions *see* access permissions
PPD 552
print using lpr 243–245
recover using testdisk 231
reference, ambiguous *1232*
remove using rm 133, 231
rename using mv 195, 245
rotate *1270*
search for using locate 264
setuid 598
sharing model 316

file, *continued*
size, display using ls 200
size, displaying easily readable 155
software package containing, search for 514, 530, 1179
sort using sort 247–249
sparse *1274*
special *see* **/dev** directory; device, files
standard 197–199
startup 188, 335–339, *1274*
tar *257*
temporary 1034, 1050
terminal 160
trash, moving to 109
truncating 591
type of, display using ls 200
utilities 280t
wiping 482
File Browser window *108*
file utility 237, 379, 492, 592
filename *189*, *1247*
/ (root) 189
ambiguous references *173*
basename *189*, *1235*
case-sensitivity 27, 187
change using mv 245
characters allowed in 186
completion *1247*
conventions 27
device *1244*
extensions 187, 188t, *1247*
extensions, remove using an event modifier 392
generation 14, 173–177, *418*, *1247*
hidden *188*, *1251*
length 186, 187, 458, 1034
root directory (**/**) 189
simple 157, *189*, *1272*
temporary 1034, 1050
typeface in this book 27
unique 1034, 1050
Windows 187
filesystem *39*, *184*, 497t, *1247*
access, speed up using tune2fs 505
autofs 864
bootable flag 85
create using mkfs 457
create using mkfs.xfs 506
demand mounting *864*
devfs 494
ext2/ext3/ext4 497, 504
filename length 458
free list 493, *1249*
hierarchy *184*
Hierarchy Standard, Linux (FHS) 12
independence 41
integrity check *503*
journaling 497, *505*, *1255*
mount
automatically 864
on demand *864*
point 40, *40*, 85, *499*, 865, *1260*
remote 848–852
remount option 207
table 502
using mount 499–501
naming 41
proc 490
RAID *see* RAID
remote *1269*
remount 207
reorganize using xfs_fsr 507
repair using xfs_repair 506
repairing 443
Standard, Linux (FSSTND) 197
structure 12
superblock *1275*
swap 42, 491
sys 492, 495
tune using tune2fs 504–505
unmount using umount 501
XFS 498, 506–507
Filesystem, Standard, Linux (FSSTND) 12
Filesystems Hierarchy Standard, Linux (FHS) 12
filling *1248*
filters *18*, *169*, *1248*
find utility 237–240, 571, 592, 614, 1016
findmnt utility 499
finger utility 321
fingerd daemon 321
fingerprint *627*, *644*, *see also* randomart image
FIOS 289
firewall *294*, *924*, 924–948, *1248*, *see also* gufw; iptables; ufw; "firewall setup" under the protocol you are running (e.g., NFS, firewall setup)
firmware *1248*
Flashback desktop, GNOME 103–104
FLOSS *7*
focus, desktop *1248*
focus, input 102
folder *108*, *see also* directory
font, antialiasing *1233*
footer *1248*
for control structure 1019–1021
for...in control structure 1017–1020
foreground *171*, *1248*
foreground versus background 171
fork *379*, *1248*
fork() system call 379, 380, 381, 996
.forward file 796
FOSS *7*
FQDN 306, *317*, *894*, *1248*
FQDN, display using hostname 227
fragmentation, disk 590
frame *26*
frame, network *1248*
free beer 2
free list, filesystem 493, *1249*
free software *7*
free space, disk *38*, 591, *1249*
Free Standards Group (FSG) 197
free utility 261
freedesktop.org group 18
fsck utility 497, 503
FSG (Free Standards Group) 197
FSSTND (Linux Filesystem Standard) 12, 197
fstab file 500, 502, 849, 853

FTP
see also FTP clients; **vsftpd**
ASCII transfer mode 760
binary transfer mode 760
debugging 763
ftp directory 765
ftp utility 754, 761–764
ftpd daemon 321
ftpusers file 777
ip_conntrack_ftp module (iptables) 942
JumpStart: downloading files using ftp 756
JumpStart: starting a **vsftpd** FTP server 765
lftp client 725
more information 755
PASV (passive) connection 755, *1264*
PORT (active) connections 755
prerequisites 756, 764
pub directory 760
security 754, 760, 765, 770
sftp client 725
site *754*
FTP clients
see also FTP; **vsftpd**
anonymous login 759
automatic login 760
basic commands 756
prerequisites 756
tutorial 757–759
using 759–764
ftp directory 765, 769
ftp file 765, 769
ftp utility 754, 761–764
ftpd daemon 321
ftpusers file 777
full backup *568*
full duplex *1249*
full regular expression *see* regular expression, extended
full-duplex network *291*
fully qualified domain name *see* FQDN
function keys, Boot menu 73
functions *1249*
bash 402–404, 1040, 1061–1062
count_down 1062
makepath 1088
mycp 1040
Python 1107, 1107t
shell *1272*
fuser utility 501

G

games directory 198
gateway *1249*
network *293*
proxy 322
gawk utility 1168
gcc *see* C programming language
GCOS *see* GECOS
gdbm database *1243*
–ge relational operator 1079
gear button 108
GECOS 830, *1249*
generic operating system 9
getfacl utility 207–211
gethostbyname() system call 903
getopts builtin 1074–1076
getpwnam() function 815
getpwuid() function 815
getty utility 380, 440
GFS filesystem 497
gibibyte *42*, *1249*
GID 485, *1250*
display using id 601
passwd file, in 486
.gif filename extension 188
gigabyte *42*, *1249*
gksudo utility 604
global variable *see* environment variable
globbing *173*, *418*
glyph *1249*
GMT *see* UTC
GNOME 17
Boxes 690
desktop, Flashback 103–104
desktop, installing 125
desktop *see also* desktop
Flashback desktop 103–104
gnome-terminal utility 126
GTK 17
Nautilus *see* Nautilus
object *see* object
panel *see* panel
Ubuntu derivative 38
workspace *see* workspace
gnome-boxes utility 690
gnome-disks partition editor 88–91
gnome-terminal utility 126
GNU
Configure and Build System 534
General Public License (GPL) *5*
Privacy Guard *see* GPG
GnuPG *see* GPG
Google Cloud Platform 703
gopher utility 325
GPG 641–656
armor 645
ASCII armor 645
encryption 641–656
file, encrypting and decrypting 649
file, signing and encrypting 650
fingerprint *644*
gpg utility 641–656
key pair, creating 641
keyring 634, 642
keyserver 652
OpenPGP Message Format 641
public key, signing 651
public keys, exporting and importing 644
public keyserver 644
seahorse utility 641
signing and verifying a file 646
web of trust 641
gpg utility 641–656, *see also* GPG
GPL (GNU General Public License) *5*
gpm daemon 321
GPT (GUID Partition Table) *445*
graphical display *1249*
graphical installation 61–67
see also installation
installer language 63
keyboard layout 65
keyboard, using 62
mouse, using 61
partitioning 63
partitioning, guided 65
time zone 65
ubiquity 61–67
user, first 66

graphical user interface *see* GUI
grave accent *see* command, substitution
grep utility 168, 240–242, 256, 1024, 1036
group 484
  access permission *199*
  add a user to a 567
  add using groupadd 567
  adm *596*
  admin *596*
  change using chgrp 203
  display name using ls 200
  file assigned to 485
  group file 484, 567
  ID *see* GID
  libvirtd 692, 698
  lpadmin *596*
  modify using groupmod 567
  password 484
  remove using groupdel 567
  sudo *596*, 604, 609
  user private 485
  user, display using groups 485
  users *1249*
  wheel and su 625
  windows *1249*
  www-data (Apache) 953
group file 484, 567
groupadd utility 567
groupdel utility 567
groupmod utility 567
groups utility 485
GRUB 444–450
  configuring 445–448
  GPT (GUID Partition Table) *445*
  grub file 446–448
  grub.cfg file 449
  grub.d directory 448
  grub-install utility 450
  grub-mkconfig utility 449–450
  hidden timeout *447*
  MBR (master boot record) 82, *445*, 450
  menu 451–453
  menu.lst file 446
GRUB 2 *see* GRUB
grub file 446–448
grub.cfg file 449
grub.d directory 448
grub-install utility 450
grub-mkconfig utility 449–450
gssd daemon 848
–gt relational operator 1079
GTK 17
guest, VM *688*
gufw 927–931
  Advanced tab 930
  Preconfigured tab 928
  rules, adding 928
  Simple tab 930
GUI *33*, 33, *1250*
  check *see* GUI, tick
  check box *1239*
  check mark *see* GUI, tick
  combo box *1240*
  dialog box *1244*
  drag *1245*
  drag-and-drop *1245*
  drop-down list *1245*
  list box *see* GUI, drop-down list
  radio button *1267*
  root privileges and 604
  scrollbar *1271*
  slider *1272*
  spin box *1274*
  spinner *see* GUI, spin box
  text box *1277*
  thumb *1277*
  tick *1277*
  tick box *see* GUI, check box
  tooltip *1278*
  WYSIWYG *1282*
  X Window System 16
GUID Partition Table *445*
guided partitioning 41, 65, *65*
gunzip utility 256, 260
.gz filename extension 188
gzip utility 253–257, 260

H

–h option 140, 155
hacker *1250*
half duplex *1250*
half-duplex network *291*
halt utility 441
hang up signal 1069
hard disk *see* disk
hard links *see* links, hard
hardware
  installation requirements 30
  list using lshw 457, *665*
  PCI devices, list using lspci *665*
  USB devices, list using lsusb 666
hash *632*, *1250*
  table *382*, *1250*
  value *632*
hash function *632*, 632–637
  digest *632*
  integrity versus authenticity 634
  MD5 *1258*
  md5sum utility 633
  message digest *632*
  one-way *1263*
  salt 634
  SHA1 *1272*
  sha1sum utility 633
  SHA2 *1272*
  sha224sum utility 632, 633
  sha256sum utility 633
  sha384sum utility 633
  sha512sum utility 633
  signing a 636
hash utility 382
hashbang *344*
HDB (Hash Database) 831
head utility 243
header, document *1250*
help
  bash command 141
  documentation 118–121, 135–142
  error messages 119
  GNU manuals 120, 1172
  –h option 140
  ––help option 140, 154
  –help option 140
  HOWTOs 121
  info utility 137–140
  Internet 119
  Linux Documentation Project 120
  local 141
  log files 1172
  mailing lists 1175t

man pages 135–137
newsgroups 476, 1174
obtaining 118–121, 135–142, 476, 1173–1175
problem solving 1172
security *656*
system manuals 135–140
Ubuntu Web site 119
window, Ubuntu 118
––help option 140, 154
–help option 140
Here document control structure 1036–1038, *1250*
hesiod *1251*
heterogeneous *1251*
hexadecimal number *1251*
hfs filesystem 498
hidden filenames *188*, *1251*
hidden filenames, display using ls 175, 229
hidden timeout, GRUB *447*
hierarchy *1251*
hierarchy, filesystem 184
hinting, subpixel *1275*
HISTFILE variable 383
HISTFILESIZE variable 383
history *1251, see also* bash history
history builtin 383
HISTSIZE variable 383
/home directory 44, 198
home directory 151, 364, *1252*
~, shorthand for 190, 365
automount 863
export using NFS 673
passwd file and 487
startup files *188*
working directory versus 194
HOME variable 365
host *688*
address 302, *304*
based trust 308
key, OpenSSH 631
nickname 306
specifying 461t
trusted 308
host utility 313, 900, 903
hostname 306–307
characters allowed in 894
symbolic 318
hostname file 485
hostname utility 227, 307
hostnamectl utility 227
hosts file 306, 485
hosts.allow file 464, 616–617
hosts.deny file 464, 616–617
hosts.equiv file 308
hotplug system 495
hot-swap *45*
hover *104*, *1252*
HOWTOs 121
hpfs filesystem 498
hping utility 657
.htaccess file 961, 967, 982, 994
.htm filename extension 966
HTML 325, *1252*
html directory 957, 961
.html filename extension 966
htop utility 578
.htpasswd file 994
htpasswd file 959
HTTP protocol *1252*
HTTPS protocol 992
https:// URI prefix 992
hub *291*, 662, *1252*
human-readable option 155
humor 7, 845, *1273*
HUP signal 1069
hypertext *325*, *1252*
Hypertext Markup Language *see* HTML
Hypertext Transfer Protocol *see* HTTP
hypervisor *688*

I

I/O device *see* device
i18n *374*, 1252
IaaS 703
IANA (Internet Assigned Numbers Authority) 319, *1252*
ICANN (Internet Corporation for Assigned Names and Numbers) 298
ICMP packet 311, 924, 941, 943, *1252*
icmp_seq 312
icon *1252*
iconify *1253*
ID string, X Window System 474
id utility 601, 606
id_dsa file 716
id_dsa.pub file 716
id_ecdsa file 716, 729
id_ecdsa.pub file 716, 729
id_ed25519 file 716
id_ed25519.pub file 716
id_rsa file 716
id_rsa.pub file 716
IDEA encryption 629
idmapd daemon 848
IDSs 658
if...then control structure 1005–1008
if...then...elif control structure 1011–1016
if...then...else control structure 1009–1011
IFS variable 369–370
ignored window *1253*
image *see* Desktop Image; Server Image; Netboot Image
IMAP server 807
in. prefix (daemon) 319
in.fingerd daemon 321
in-addr.arpa domain 902
include directory 198
increment operators 1083
incremental backup *568*
indentation *see* indention
indention *1253*
inequality operator 1082
info directory 199
info utility 137–140, 178
infrastructure mode, wireless 664
inheritance, bash 1054
init daemon 380, *see also* Upstart
init directory 429, 433
init scripts 435
init.d directory 435
initctl utility 429
inittab file 435, 486
inodes *493*, *1253*
alter using mv 493
create reference using ln 493
delete reference using rm 493
display using ls 214
locate using find 1016

input *1253*
input focus 102, *102*
input, standard *see* standard input
input/output device *see* device
.inputrc file 396
INPUTRC variable 396
insmod utility 444
installation 59–93
  see also installation disk; installation image; installation medium; ubiquity
  BIOS, set to boot from DVD or flash drive 32
  boot command line 74
  boot commands 76
  boot menu, displaying 57
  Boot menus 72–75
  boot parameters 75–77
  booting 56
  CD/DVD, software, installing from 122
  CMOS setup 32
  computer *1253*
  CPU requirements 31
  databases, initializing 70
  Desktop Boot menu 72
  display, configuring 116
  dual-boot system 91–93
  Expert mode 75
  failure 56
  function keys 73
  gnome-disks partition editor 88–91
  hardware requirements 30
  installation image, from an 59–61
  ISO image file, verifying 633
  language overlay 57, 71, 73
  live session, from a 60
  MD5SUMS file 633
  medium *36*
  medium, testing 58
  minimal system 74
  Netboot Boot menu 75
  old/slow systems 32
  partitioning, manual 42
  partitioning, manual (textual) 82–87
  planning 30–46
  processor architecture 32
  RAID 45–46, 86
  RAM (memory) requirements 31
  RAM (memory), test using memtest86+ 72
  Server Boot menu 72
  SHA1SUMS file 51, 633
  SHA256SUMS file 633
  textual installer 71, 78–82
  ubiquity installer *see* ubiquity
  ubiquity partition editor 67–70
  UEFI 32
  unetbootin utility 52
  updating the system 70
  verifying an ISO image file 633
  virtual consoles 61
installation disk
  see also installation; installation image; installation medium; ubiquity
  formatting 38
  free space *38*
  guided partitioning 41
  partition
    delete using gnome-disks 91
    display using gnome-disks 88
    set up 38
    set up using gnome-disks 88–91
    setup, guided 41
  setup 63
  space requirements 32
installation image *36*
  see also installation; installation disk; installation medium; ubiquity
  checking for defects 72
  Desktop 36, 71
  downloading 47–50
  function keys 73
  installing 61–67
  Netboot 37, 71
  Server 37, 71
  verifying 51
installation medium
  see also installation; installation disk; installation image; ubiquity
  BIOS, set to boot from 32
  burning a DVD 52
  checking for defects 72
  testing 58
  writing to a USB flash drive 52
INT signal 1069
Integrated Services Digital Network *see* ISDN
integrity versus authenticity 634
interactive *1253*
interface *1253*
  character-based *see* command line; textual, interface
  command-line *see* command line; textual, interface
  graphical user *see* GUI
  pseudographical *33*, 126
  textual *see* command line; textual, interface
  user *1279*
interface-local *301*
internal field separator *see* **IFS** variable
International Organization for Standardization *see* ISO
internationalization *374*
Internet 286, *1253*
  Assigned Numbers Authority *see* IANA
  browsers 326
  connection sharing 945–948
  Control Message Protocol *see* ICMP
  look up site information using whois 314
  multiple clients on a single connection 946
  multiple servers on a single connection 948
  netiquette *1261*
  Printing Protocol *see* IPP
  Protocol Security *see* IPSec
  Protocol *see* IP; TCP/IP
  search engines *326*
  service provider *see* ISP
  URI *1279*
  URL *326*, *1279*
internet (*lowercase "eye"*) *1253*
internetwork *286*
InterNIC 314

interprocess communication 18, 495, *495*
interrupt key *130*
intranet *287*, *1254*
intrusion detection system *see* IDS
invisible files *see* hidden filenames
IP *1254*
  see also IP address; IPv6
  masquerading 933, 943, 946
  multicast *see* multicast
  protocol model 296
  spoofing *1254*
  TTL header field *312*
  version 6 *see* IPv6
IP address *1254*
  client, specifying 461t
  computations 305t
  dynamic 304, 670
  loopback service *485*
  representation 298
  static 304, 467, 670
ip utility 468
ip6.int domain 902
ip6tables-restore utility 944
ip6tables-save utility 944
IPC *1254*
ipchains utility 932
IPP 540
iptables 932–948
  see also gufw; iptables rules; ufw
  chain *932*
  chain policy 938
  classifiers *932*
  command line 936–937
  commands 938
  connection tracking 934, 942
  conntrack module 934
  counters 944
  display criteria 940
  DNAT targets 933, 942
  Filter table 933
  ICMP packet 941, 943
  Internet connection sharing 945–948
  IP masquerading 946
  ip_conntrack_ftp module 942
  ip6tables-restore utility 944
  ip6tables-save utility 944
  ipchains utility 932
  iptables-restore utility 944
  iptables-save utility 944
  jumps 937
  Mangle table 934
  masquerade *1258*
  MASQUERADE targets 933, 943
  match criteria 936
  match extensions 940–942
  matches 932
  more information 935
  NAT table 933
  NAT, running from 945–948
  netfilter 932
  network packet 934
  packet match criteria 936, 939
  ping utility 943
  policy command 938
  prerequisites 935
  protocols file 940
  resetting rules 935
  router 945, 948
  services file 941
  SNAT targets 933, 943
  state machine 934, 942
  targets 932, 933, 937, 942–944
iptables rules 932
  building a set of 937
  example 932
  match criteria 937
  number 937
  saving 944
  specification 936
iptables-restore utility 944
iptables-save utility 944
IPv4 298
IPv6 299–302, *1254*
  see also IP
  address 300
  address records, DNS 302, 898
  DHCP 302
  interface-local *301*
  link-local addresses *301*
  multihoming *301*
  ping6 312
  traceroute6 313
  tunnel brokers 299
IRC, Ubuntu channels 120
irqpoll boot parameter 77
is_regfile shell script 1007
isatty() system call 160
ISDN 289, *1254*
ISO *1254*
ISO image file *47*
ISO image file, verifying 633
ISO protocol model 297
ISO9660 filesystem 498, *1255*
ISP *1255*
iterable *1112*, *1117*, *1118*

J

jffs2 filesystem 498
job *171*, *352*
  control 15, 171, 352–355, *1255*
  jobs builtin 172, 352
  number 171
  number, display using jobs 172
  suspend using CONTROL-Z 172
  Upstart *428*, 429, 430–432
jobs builtin 131
john utility (John the Ripper) 615, 657
join *1137*
journaling filesystem 497, 505, *1255*
Joy, Bill 1237
JPEG *1255*
.jpeg filename extension 188, *1255*
.jpg filename extension 188, *1255*
JumpStarts *see* the JumpStart index (1285)
justify *1255*

K

K&R 11
KDE 17
  desktop 37, 125
  Qt toolkit 17
Kerberos *1255*
kernel 7, *1255*
  see also Linux
  booting 438
  command execution environment *1054*

kernel, *continued*
depmod utility 444
insmod utility 444
loadable module *444*, *1257*
lsmod utility 444
messages, display using dmesg 439, 454
modinfo utility 444
modprobe utility 444
module *444*, *1257*
modules file 444
modules, tools for working with 444t
packages, list installed using dpkg 449
packages, remove using aptitude 449
packet filtering *see* gufw; iptables; ufw
proc pseudofilesystem 490
programming interface 11
rmmod utility 444
space *1255*
version, display using uname 449
kernelspace *1255*
Kernighan & Ritchie 11
key space *628*
keyboard *1255*
keyboard as standard input 160
keyboard layout, graphical installation 65
keyring 634, 642
keys
BACKSPACE (erase character) 130
binding *1255*
CONTROL *27*
CONTROL-\ (quit) 131
CONTROL-C (copy) 107
CONTROL-C (interrupt) 130
CONTROL-D (EOF) 128, 161
CONTROL-H (erase character) 130, 156
CONTROL-U (line kill) 130, 156
CONTROL-V (paste) 107
CONTROL-W (erase word) 130, 156
CONTROL-X (cut) 107
CONTROL-X (line kill) 130
CONTROL-Z (suspend) 130, 131, 172
cryptographic *627*
DEL (erase character) 130
end line 27
ENTER 27
erase *129*
interrupt *130*
line kill 130, 156
META *1259*
NEWLINE 27
RETURN 27, 156
typeface 27
keyserver 652
keyword variables 359
keywords, search for using apropos 137
kill builtin 131, 172, 455–457, 1070, 1072
kill line key (CONTROL-U) *130*, 156
kill process 501
KILL signal 131, 1070
killall utility 457
kilo- *1256*
known_hosts file 716, 718
Korn, David 335, 1256
Korn Shell 335, *1256*
ksh shell 335, *1256*
Kubuntu 37
KVM 690
KVM kernel modules 692
Kylin desktop 38

L

l10n *374*, 1256
LAMP 37, 675, 1136
LAN 290, *1256*
more information 662
setting up 662–666
LANG variable 374, 953, 1176
language, procedural *1086*
language, programming 1174t
large number *1256*
last utility 623
launchd daemon 427
Launchpad 511, 657
LBA addressing mode, disk 445
LC_ALL variable 374
LC_COLLATE variable 374
LC_CTYPE variable 374
LC_MESSAGES variable 374
LC_NUMERIC variable 374
LDAP 830–839, *1256*
BDB 831
Berkeley DB 831
CN *832*
Common Name *832*
DB_CONFIG file 834
DC *831*
directory *830*
DN *831*
DSA *831*
DSE *831*
entry *831*
firewall setup 834
GECOS 830
HDB (Hash Database) 831
ldapadd utility 835
ldapdelete utility 835
ldapmodify utility 834
ldapsearch utility 834, 835, 836
LDIF *832*, 832
makedbm utility 826
objectClass 832
OpenLDAP 831
passwd utility 821
prerequisites 833
RDN *831*
RTC *831*
schema directory 831
server, test 834
setting up a server 833
slapcat utility 835
slapd daemon 833
slapd service, starting 833
Sleepycat Berkeley Database 831
Thunderbird client 838
ldapadd utility 835
ldapdelete utility 835
ldapmodify utility 834
ldapsearch utility 834, 835, 836
ldd utility 616, 618
.ldif filename extension 832
LDIF, LDAP *832*
–le relational operator 1079
leaf *1256*
least privilege *597*, *1256*
left-click 105

left-handed mouse 116, 475
less utility 136, 169, 228, 1011
let builtin 416, 1078
lftp utility *725*
lftp.conf file 725
.lftprc file 725
/lib directory 198
lib directory 198
/lib64 directory 198
libexec directory 492
libraries called by executable, list using ldd 616
library *1256*
library, **libwrap** 616
libvirt 690
libvirtd group 692, 698
libwrap library 616
LightDM display manager 440
lightdm.conf file 473
lightweight desktop *see* LXDE desktop
Lightweight Directory Access Protocol *see* LDAP
line kill key (CONTROL-U) *130*, 156
line numbers, display using cat 225
Line Printer Daemon *see* **lpd** daemon
LINES variable 1035
link-local addresses (IPv6) *301*
links *13*, *211*, 211–217, *1256*
  alternatives directory 483
  delete using rm 216, 493
  display using ls 214
  exporting 857
  find using **lnks** 1013
  hard 212–214, 493, *1250*, *1256*
  hard versus symbolic 212, 214
  hard, create using ln 212, 493
  hypertext *325*
  inode 493
  number of, display using ls 200
  point-to-point *1265*
  remove using rm 216
  soft *see* links, symbolic
  symbolic *214*, 494, *1256*, *1276*
    as special files 494
    cd and 216
    create using ln 215
    dereference *1243*
    versus hard 212, 214
  symlinks *see* links, symbolic
  utility names 483
links utility 326
Linux
  see also kernel
  benefits 7–9
  distribution 6
  documentation 118–121, 135–142
  Documentation Project 120
  FHS (Filesystem Hierarchy Standard) 12, 197
  file namespace *40*
  Foundation 197
  FSSTND (Filesystem Standard) 12, 197
  history 2–9
  LSB (Linux Standard Base) 197
  manual sections 136
  newsgroups 476, 1174
  overview 11–19
  PAM *see* PAM
  Pluggable Authentication Modules *see* PAM
  standards 8
  Terminal Server Project 845
  UNIX heritage 2
linux terminal name 1175
linux-vdso.so.1 file 618
list box *see* drop-down list
list operator *see* character, class
lists *170*
lists directory (APT) 518
lists, Python 1109–1113
listserv 307
live session *36*, *59*, 60
live session, RAM usage 36
ln utility 212, 215, 493
ln utility versus cp 213
lnks shell script 1013
load average, display using uptime 261
load average, display using w 262
loadable modules *444*, *1257*
local area network *see* LAN
local builtin 403, 1062
/local directory 44
local directory 198, 534
local variables 382
local.cf file 799
locale 374–376, *1257*
  Apache 953
  C 375
  i18n *374*
  internationalization *374*
  l10n *374*
  LANG variable 374
  LC_ variables, setting 376
  LC_ALL variable 374
  LC_COLLATE variable 374
  LC_CTYPE variable 374
  LC_MESSAGES variable 374
  LC_NUMERIC variable 374
  locale utility 375–376
  locales directory 376
  localization *374*
  setlocale() system call 375
locale utility 375–376
locales directory 376
localhost 306
locality, process 1054
localization *374*
localtime file 379
locate database 70
locate utility 70, 264
lockd daemon 321, 848
locking the **root** account 613
locktty shell script 1026
log
  analyze using swatch 659
  DNS 919
  email 786
  files
    checking 590
    obtain help using 1172
    rotate using logrotate 582–584
  FTP 774
  in *see* login
  log directory 492, 585
  machine 589, 589t
  OpenSSH 735
  out 441, *1257*
log directory 199, 1172
logging in *1257*
  see also login
  problems 142
  remotely 128

logging out 128
logical
  evaluation 1079
  expressions *1257*
  operators *see* Boolean operators
Logical Volume Manager *see* LVM
login 440
  automatic using OpenSSH 728–730
  name *see* usernames
  options 99
  problems 587
  prompt 440
  root *599*, *1270*
  screen 99
  shell *336*, *1257*
  user, controlling *599*
.login file *1257*
Login incorrect error message 440
login utility 380, 440
login.defs file 486, 566
.logout file *1257*
logresolve utility 973
logrotate utility *582–584*
logrotate.conf file *582–584*
logrotate.d directory *582–584*
logvsftpd.log file 777
logwatch utility *590*
loopback service *485*
lossless *253*
lost+found directory 443, 480
lp utility *557*
lpadmin group *596*
lpadmin utility *552–555*
lpd daemon 540
LPI certification 1183–1229
lpinfo utility *552*
lpq utility 245, *557*
LPR line printer system 540
lpr utility 168, 243–245, *557*
lprm utility 245, *557*
lpstat utility *557*
ls utility 132, 153, 199, 229
LSB (Linux Standard Base) 197
lsb_release utility 447
lsblk utility *665*
lshw utility 457, *665*
lsmod utility 444, 692
lsof utility *589*
lspci utility *665*
lsusb utility *666*
–lt relational operator 1079
LTS release *35*
Lubuntu 37
LV *see* LVM, LV
LVM 46
  LV *46*
  PV *46*
  VG *46*
LXDE desktop 37, 125
lynx utility 326

M

MAAS 72
MAC address 302, 468, *1257*
Mac processor architecture 33
mac2unix utility 268
machine collating sequence *1258*
machine log *589*, *589t*
machine name, display using hostname 227
macro *1258*
magic file 492, 960, 1258
magic number *492*, *1258*
mail
  see also **postfix**
  aliases 483, 794
  aliases file 794
  checking **root**'s 590
  dovecot 807
  Evolution 800
  exim 783
  firewall setup 784, 808
  forwarding 794–796
  IMAP server (Dovecot) 807
  list server 307
  log 786
  mail directory 794
  MAIL variable 366
  mail.err file 590
  mail.log file 590, 786
  mailbox *366*
  MAILCHECK variable 367
  maildir format 781
  mailing list 804–807
  Mailman 805–807
  MAILPATH variable 367
  mailq utility 786
  mbox format 781
  MDA 780, *1258*
  more information 783
  MTA 780, *1260*
  MUA 780, *1260*
  newaliases utility *795*
  POP3 server (**dovecot**) 807
  postmaster *590*
  Qmail 783
  sendmail 783
  sendmail daemon 322
  SMTP 780
  spam *see* spam
  SpamAssassin *see* SpamAssassin
  SquirrelMail 802–804
  Thunderbird 800
  UCE *see* spam
  Webmail 801–804
mail directory 794
MAIL variable 366
mail.err file 590
mail.log file 590, 786
mailbox *366*
MAILCHECK variable 367
maildir format 781
mailing list 804–807
Mailman 805–807
mailman service, starting 806
mailname file 787, 789
MAILPATH variable 367
mailq utility 786
main memory *1258*
  see memory; RAM
main software package category 516
main.cf file 787, 789
mainframe computer *9*
mainframe model 315
major device number *496*, *1258*
makedbm utility 826
makepath function 1088
makewhatis database 137
Mallory (placeholder name) 627
MAN *293*, *1258*
man directory 199

man utility 135–137
manager, file *see* Nautilus
mandb database 70, 137
man-in-the-middle attack *631*, *1258*
manuals *see* help
map files 865
mapping a share *871*
MariaDB 1136–1155
  see also SQL
  adding a user 1144
  adding data 1147
  column *1137*
  comments 1138
  creating a database 1143
  creating a table 1145
  database *1137*
  datatypes 1138
  history 1136
  installing 1140
  joins *1137*, 1152–1155
  logging in 1145
  .my.cnf file 1142
  MySQL compatibility 1136
  mysql_history file 1142
  mysqldump utility 1150
  options 1140
  prompt 1138
  relational database management system *1137*
  retrieving data 1148
  root password 1141
  row *1137*
  SELECT statement 1148
  SQL *1137*
  table *1137*
  terminology 1137
mask *see* network, mask
masquerading, IP 933, 943, 946, *1258*
Massachusetts Institute of Technology *see* MIT
master.cf file 794
mawk utility 1021
mbox format 781
MBR (master boot record) 82, *445*, 450
MD device 86
MD5 *1258*
md5sum utility 633
MD5SUMS file 633
MDA 780, *1258*
mebibyte *42*, *1259*
megabyte *1259*
memory
  see also RAM
  free, allocating to buffers 261
  information about, display via **proc** 491
  main *1258*
  memtest86+ utility 72, 448
  paging *491*
  shared 483
  test using memtest86+ 72
  usage, display using free 261
  virtual and swap space 491
  virtual, report on using vmstat 576
memtest86+ utility 72, 448
menu *1259*
  bash 1034
  panel *see* panel
  shell script 1030
  Window Operations 106
menu.lst file 446
menubar *106*
merge *1259*
message
  see also error messages; usage messages
  daemon 492, *585–586*
  messages directory 492
  of the day *see* **motd** file
  rsyslog.conf file 585–586
  rsyslogd daemon 585–586
  security 492
  system 492
  usage *see* usage messages
message digest *632*
Message Digest 5 *see* MD5
messages file 1172
META characters, display using cat 225
META key *1259*
metabit *1233*
metacharacters *1259, see also* special characters
metadata *1259*
metapackage *519*
metropolitan area network *293*, *1258*
microprocessor 10
Microsoft Azure 703
Microsoft Windows *see* Windows
middle mouse button 107
MIME 112, *1259*
mingetty utility 380
minicomputer *9*
mini-HOWTOs 121
Minimal edition 37
Minimal Image 37
minimal system 74
minimize window *1260*
MINIX *5*
minix filesystem 498
minor device number *496*, *1260*
mirror *748*
mirrors, Ubuntu 49
mistakes, correct typing 129
MIT, Project Athena 15
MIT, X Consortium 15
MITM *see* man-in-the-middle attack
mkdir utility 192–193
mkfifo utility 495
mkfs utility 457, 480
mkfs.xfs utility 506
mklost+found utility 480
mkswap utility 491
mm_cfg.py file (Mailman) 806
/mnt directory 198
mod_perl.so file 959
modem *1260*
modem, cable *1238*
modinfo utility 444
modprobe utility 444
mods-available directory 957, *959*
mods-enabled directory 957
module, kernel *444*
modules directory 198, *959*
modules file 444
moduli file 715
monitor, configuring 116
monthly file 575
more utility 228
Mosaic Web browser 325

most utility 228
motd file 486
motherboard *1260*
mount *1260*
  automatic 864, *1234*
  CD-ROM 500
  filesystem using mount 499–501
  find mounted filesystems using findmnt 499
  point 40, *40*, 85, *499*, 865, *1260*
  remote filesystems 848–852
  remount option 207
  table 486, 502
mount utility 207, 486, 499–501, 848, 849–852, 876
mountd daemon 321
mounts file 486
mouse *1260*
  click 105
  double-click timeout 116
  left-handed 116, 475
  middle button 107
  mouseover *1260*
  pointer *1260*
  pointer, hover *104*, *1252*
  preferences, setting 116
  remapping buttons 116, 475
  right-click 106
  right-handed 475
  wheel 475
mouseover *1260*
mpage utility 554
msdos filesystem 498
MTA 780, *1260*
mtab file 486
mtr utility 313
MUA 780, *1260*
multiboot specification *1260*
multicast *1260*
multidisk device 86
multihoming *301*
multimedia Ubuntu 38
multiplication operator 1082
Multipurpose Internet Mail Extension *see* MIME
multitasking *12*, *1261*
multiuser *12*, *1261*
multiuser mode 439
multiverse software package category 516
mv utility 195, 196, 245, 493
MX records, DNS 899
.my.cnf file 1142
mycp function 1040
mysql_history file 1142
mysqldump utility 1150
Mythbuntu 38

N

name
  command 152
  daemon 319
  domain *see* domain, name
  login *see* username
  servers 316, 318
named daemon 905
named directory 917
named pipe *495*
named.conf file 907, 911
named.conf.options file 912
nameserver *892*
namespace *40*, *1126*, *1261*
nano utility 277–280
NAT *1261*
  routers and 663
  running from iptables 945–948
  table, iptables 933
National Center for Supercomputer Applications 325
Nautilus 108–113
  access permissions 111
  File Browser window *108*
  file, open with 110
  Open With selection 112
  terminal emulator, open with 110
NBT *1261*
ncpfs filesystem 498
ndbm database *1243*
NdisWrapper 664
–ne relational operator 1079
negation operator 1082
nessus utility 658
net use utility (Windows) 888
net utility 870
net view utility (Windows) 888
NetBIOS *1261*
netboot 846, *1261*
Netboot edition 37
Netboot Image 71
netiquette *1261*
.netrc file 760
Netscape 6, 325
Netscape BugSplat bug tracking system 536
network
  see also IP address; protocols; wireless
  address *304*, *1261*
  address, @ in 307
  analyze using wireshark 659
  ARP (Address Resolution Protocol) 303
  arp utility 303
  boot *1261*
  bottleneck, find using traceroute 312
  bridge *292*, *1237*
  broadcast 288, *1237*
  broadcast address *1237*
  cables *see* cables
  client/server model 316
  collision domain *292*
  concentrator *see* network, hub
  configure using NetworkManager 667–672
  connection, test using ping 311
  daemons *286*, 319
  datagram *1243*
  DNS *see* DNS
  drivers 664
  duplex *291*
  Ethernet 290, *1247*
  extranet *287*, *1247*
  FDDI *1247*
  file sharing model 316
  firewall *see* firewall
  frame *1248*
  full-duplex *291*
  gateway *293*, *1249*
  half-duplex *291*
  hops *312*
  host address 302, *304*
  hostname, FQDN *see* FQDN
  hosts file 306

hub *291*, 662, *1252*
ICMP packet *1252*
interface card *see* network, NIC
internet (*lowercase "eye"*) *1253*
Internet *see* Internet
internetwork *286*
intranet *287*
layer, IP model protocol *297*
local area *see* LAN
MAC address 302
mainframe model 315
mask *304*, 462, *1262*
metropolitan area *293*, *1258*
monitor with Cacti 674–683
multicast *1260*
nameservers 316, 318
NdisWrapper 664
NIC 302, 664
node *662*
node address *304*
number *see* network, address
packet *288*, 934, *1264*
packet filtering *1264, see also* gufw; iptables; ufw
packet sniffer *1264*
partner net *287*
point-to-point link 289
port forwarding *1266*
private address space 667, 667t, *1266*
privileged port *1266*
PTP (peer-to-peer) model 316
resolver *318*
route, display using traceroute 312
router *292*, *293*, 663, *1270*
segment *291*, *1262*
services 286, 319
setting up 662–666
sniff *1273*
sockets *495*
specifications 288t
subnet 304, *304*, *1275, see also* network, address
  addresses *1275*
  masks *304*, *1275*
  numbers *1275*
  specifying 462, 462t
switch 289, *291*, 662, *1262*
token ring *1278*
topologies *288*
topology, shared *1272*
trusted hosts 308
tunneling *1278*
UDP *1278*
unicast *1279*
VPN *287*, *1280*
WAN *see* WAN
wide area *see* WAN
Wi-Fi *1281, see also* wireless
wireless *see* wireless
Network Address Translation *see* NAT
Network File System *see* NFS
Network Information Service *see* NIS
Network Time Protocol *see* NTP
NetworkManager
  applet 667
  daemon 667
  nmcli utility 671
  static IP address 670
newaliases utility *795*
NEWLINE control operator 347
NEWLINE key 27
NEWLINE, quote using a backslash 351
newlist utility 805
news, Internet *see* newsgroups
newsgroups 476, 1174
NFS 844–847, *1262*
  all_squash option 860
  attribute caching options 850
  block size 852
  data flow 844
  df utility 846
  directory, shared 673
  error handling options 851
  error messages 851, 862
  etab file 861
  exportfs utility 856, 861
  exporting device files 857
  exporting directory hierarchies 857–860
  exporting links 857
  exports file 855, 857–860, 861
  filesystem 498
  firewall setup 854
  fstab file 849, 853
  gssd daemon 848
  home directory, export using 673
  idmapd daemon 848
  JumpStart: configuring an NFS server using shares-admin 855–856
  JumpStart: mounting a remote directory hierarchy 848
  line speed, testing 852
  lockd daemon 321, 848
  more information 847
  mount utility 848, 849–852
  mountd daemon 321
  mounting remote directory hierarchies 848–852
  NFS server *xxx* not responding error message 862
  nfs-common init script 848
  nfsd daemon 321
  nfs-kernel-server init script 854
  NIS and 860
  options 858–860
    all_squash 860
    attribute caching 850
    error handling 851
    miscellaneous 851
    root_squash 860
  performance, improving 852
  prerequisites 847, 853
  rpc.gssd daemon 848
  rpc.idmapd daemon 848
  rpc.lockd daemon 848
  rpc.statd daemon 848
  rpcbind daemon 847, 854
  rpcbind service, starting 847, 854
  rquotad daemon 321
  running clients 847
  security 847, 855, 860
  server not responding error message 851
  server–server dependency 864
  setuid 848
  shares-admin utility 855–856, 858
  Stale file handle error message 862
  statd daemon 321, 848
  testing 862
  timeout 851, 852

nfsd daemon 321
NIC 302, 664, *1262*
nice utility *576*
nickname, host 306
nicknames file (NIS) 816
NIS 814–816, *1262*
  broadcast mode 817
  client, setting up 817–822
  debugging 820, 829
  defaultdomain file 818
  domain 814
  domain name 818, *1262*
  firewall setup 823
  GECOS 830
  makedbm utility 826
  Makefile file 826
  maps 815
  master servers 814
  more information 816
  NFS and 860
  nicknames file 816
  nis file 818, 823, 824, 830
  nisdomainname utility 818
  nsswitch.conf file 814
  passwd utility 821
  prerequisites 817, 823
  rpcbind service, starting 817, 823
  rpcinfo utility 820
  securenets file 826
  server, setting up 822–829
  server, specifying 819
  slave servers 814
  source files 815
  terminology 816
  testing 829
  troubleshooting 820, 829
  users, adding and removing 822
  Yellow Pages 814
  yp.conf file 819
  ypbind daemon 818, 820
  ypinit utility 828
  yppasswd utility 821–822
  yppasswdd daemon 830
  yppush utility 826
  ypserv.conf file 824
  ypwhich utility 820
  ypxfr utility 823
  ypxfrd daemon 823
nis file 818, 823, 824, 830
nmap utility 658
nmbd daemon 870, 888
nmblookup utility 888
nmcli utility 671
NNTP (Network News Transfer Protocol) *1262*
No such file or directory error message 365
noacpi boot parameter 76
noapic boot parameter 76
noapm boot parameter 76
noclobber variable 164–166
node *662*, *1262*
node address *304*
node, VM *688*
nodma boot parameter 76
nofb boot parameter 77
nolapic boot parameter 77
–nolisten tcp option (X Window System) 472
nologin file 442, 625
nologin utility 487
nologin.txt file 487
nonprinting character *1263*
nonrepudiation of origin *see* cryptography, nonrepudiation of origin
nonvolatile storage *1263*
NOT (**!**) Boolean operator 1085
nsswitch.conf file 468–471, 814, 903
ntfs filesystem 498
NTP *1263*
ntpd daemon 321
null device 166
null file 166, 481, 591, 1024
null string *1263*
number
  block *1236*
  gibibyte *1249*
  gigabyte *1249*
  hexadecimal *1251*
  kilo- *1256*
  large *1256*
  magic *492*, *1258*
  mebibyte *1259*
  megabyte *1259*
  octal *1263*
  sexillion *1272*
  tera- *1276*
  undecillion *1279*

O

–o (OR) Boolean operator 1079
.o filename extension 188
object 104, *105*
  copying 107
  cut and paste 107
  panel *see* panel
  right-click 106
  selecting 110
  trash, moving to 109
Object Properties window 111–113
objectClass, LDAP 832
octal number *1263*
octet *298*
od utility 483
old systems, installation on 32
OLDPWD variable 414
one-way hash *1263*
Open Group 15
open source 6, *1263*
Open Source Initiative *see* OSI
OpenLDAP 831
OpenPGP Message Format 641
OpenShift 703
OpenSSH 714–738, *1263*
  authentication 631, 714, 717
  authorized keys 728–730
  authorized_keys file 715, 730
  automatic login 728–730
  client, configuring 717–719
  clients 716–727
  compression 738
  config file 715
  configuration file, server 732–734
  configuration files, client 726–727
  configuration files, client and server 714–716
  debugging 723, 725, 732, 735
  encryption 631, 714
  environment file 716
  firewall setup 728

firewall, working with 738
id_dsa file 716
id_dsa.pub file 716
id_ecdsa file 716, 729
id_ecdsa.pub file 716, 729
id_ed25519 file 716
id_ed25519.pub file 716
id_rsa file 716
id_rsa.pub file 716
init script 728
JumpStart: starting an OpenSSH server 728
JumpStart: using ssh and scp to connect to an OpenSSH server 716
key files, host 728
keys 631
keys, personal, store using ssh-agent 731–732
known_hosts file 716, 718
log file 735
moduli file 715
more information 716
opening a remote shell 720
password vs. personal key authentication 730
port forwarding 735–737
prerequisites 716, 727
randomart image *730*
recommended settings, client 717
recommended settings, server 728
rhost authentication 715
rsync utility 724
running commands remotely 720–722
security 714
server, setting up 727–734
ssh_host_dsa_key file 715
ssh_host_dsa_key.pub file 715
ssh_host_ecdsa_key file 715
ssh_host_ecdsa_key.pub file 715
ssh_host_ed25519_key file 715
ssh_host_ed25519_key.pub file 715
ssh_host_rsa_key file 715
ssh_host_rsa_key.pub file 715
ssh_known_hosts file 715, 719
ssh-add utility 732
ssh-agent utility 731–732
ssh-copy-id utility 730
sshd daemon 727–734
sshd_config file 730
ssh-import-id utility 715
ssh-keygen utility 729–731
troubleshooting 735
tunneling 735–737
X11 forwarding 717, 726, 734, 736
OpenSSL 637–641
openssl utility 637–641
OpenStack 703
operand *152*
operating system *1263*
choosing 19
generic 9
proprietary 9
Operations menu, Window 106
operators
arithmetic (bash) 1081t, 1081–1085
Boolean *see* Boolean operators
list *see* character, class
logical *see* Boolean operators
redirection (bash) 342, 342t
relational 1079
remainder 1084
short-circuiting *349*
string 1080t
ternary 1084
/opt directory 44, 198
opt directory 198, 534
OPTARG variable 1074
optical, drive *1263*
OPTIND variable 1074
optional sections (in this book) 28
options *153*, *1263*
boot 75–77
command line 153–155, 405
flags, displaying (bash) 1052
OR (|) Boolean operator (extended regular expression) 1168
OR (||) Boolean operator 1079, 1084
OR (**–o**) Boolean operator 1079
OR bitwise operator (|) 1084
ordinary file *151*, *185*, *493*, *1263*
OSDL (Open Source Development Labs) 197
OSI 7
other access permission *199*
out shell script 1011
output *1264*
output, standard *see* standard output
owner access permission *199*
owner of file, display using ls 200

P

.p filename extension 1122
P2P *1264*
Paas 703
package group *519*
package management system, *510*
package *see* software packages
PackageKit 511
packet *1264*
filtering *1264*, *see also* gufw; iptables; ufw
network *see* network, packet
sequence number (**icmp_seq**) 312
sniffer *1264*
page breaks 276
pagers *136*, *228*, *1264*
paging *491*, *1264*
PAM 621–626, *1264*
auth.log file 622
authentication stack *621*
control flag keywords 624t
features 440
module type indicators 623t
more information 622
pam.d directory 621
security file 621
stack *621*
pam.d directory 621
panel 104, *105*
parameter null or not set error message 1060

parameters *358*
 see also bash parameters
 boot, modifying 75–77
 expansion (bash) 414
 positional *1044*, 1044–1049
 quiet boot 58
 shell *358*, 358–371
 special 1049–1052
 splash boot 58
parent directories 185, 186
parent process 380, *1264*
parentheses, group commands using 350
parse *156*, *410*
parted utility *579–582*
partition *38*, *1264*
 see also name of partition (e.g., **/var** [indexed under **var** directory])
 create manually (graphical) 63
 create manually (textual) 82–87
 create using gnome-disks 90
 create using parted *579–582*
 create, guided 41, *65*, *65*
 creating, about 41
 delete using gnome-disks 91
 display using gnome-disks 88
 editor, ubiquity 67–70
 encrypt using gnome-disks 90
 extended 39
 filesystem *39*
 logical 39
 naming 41
 primary 39
 RAID *see* RAID
 sizes, minimum 44t
 sizes, suggested 42
 swap 42
 table 38
 table, write using gnome-disks 89
 type 84
 UUID number 502
 work with using gnome-disks 88–91
partner net *287*
partner software package category 516
PASC (Portable Application Standards Committee) 335
passive FTP *see* FTP, PASV
passphrase *627*
passwd file 440, 486–487, 599
passwd utility 144, 821
passwords *1264*
 change using passwd 144
 change using User Accounts window 118
 choosing 143, 615
 cracking 615
 generating using pwgen 144
 group 484
 hashed 489
 John the Ripper utility 657
 locked 489
 no password 489
 passwd file 440, 486–487, 599
 root account 598
 root account and sudo 612
 root account, assigning to 613
 Samba 873
 security 143, 615
PASV FTP *see* FTP, PASV
path, search 263
PATH variable 158, 337, 365–366, 613, 614, 1033
pathnames *185*, *189*, 195, *1264*
 / trailing within 40
 / within 40
 ~ (tilde) in a 190
 absolute 157, *189*, *1232*
 completion 395
 elements *1265*
 expansion *173*, 173–177, 360, 418
 last element of *1264*
 relative 157, *190*, *1269*
PC processor architecture 33
PCI devices, list using lspci 664
pdbedit utility (Samba) 873
.pdf filename extension 188
PDF printer, setting up a virtual 542
peer, BitTorrent *531*
.pem filename extension 705
Perens, Bruce 7
period special character 1163
peripheral device *see* device
Perl, Apache, scripts called from *995*
Perl, CGI script 992
Permission denied error message 158, 343, 365
permissions *see* access permissions
persistent *1265*
PGP 641, *see also* GPG
pgrep utility 456
philosophy, UNIX 307
phish *1265*
.php filename extension 966
physical device *1265*
physical layer, IP model protocol *297*
PID *1265*
 $! variable 1051
 $$ variable 1050
 background process 171
 display using ps 172
 fg 334
 number *380*
 number 1 380, 439
pidof utility 457
pinfo utility 139
ping utility 311, 458, 852, 943
ping6 utility 311, 312
pipe, named *495*
pipe, named, create using mkfifo 495
pipeline *18*, *166*, 166–170, *1265*
 see also pipeline symbol
 precedence of 1083
pipeline symbol (|)
 see also pipeline
 continuation, implicit 1021, 1085
 extended regular expressions, in 1168
 filters, in 169
 lists, in 170
 noclobber and 165
pixel *1265*
pkexec utility 604
PKI *1265*
pkill utility 458
.pl filename extension 966
plaintext *627*, *1265*

Pluggable Authentication Module *see* PAM
plus sign (**+**) in extended regular expressions 1168
PMS (package management system) *510*
point release 35
point-to-point link 289, *1265*
Point-to-Point Protocol *see* PPP
PolicyKit 604
POP3 server (Dovecot) 807
popd builtin 357
portable *9*
portmapper 323, *1266*
ports *318*, *1265*
  connect to using telnet 310
  forwarding *1266*
  forwarding using OpenSSH 735–737
  privileged *318*
  scan for open using nmap 658
  setting serial information 458
ports.conf file 960
positional parameters *1044*, 1044–1049
POSIX 8, 335
postfix 781–796
  see also mail
  aliases 794
  aliases.db file 795
  alternatives 783
  configuration parameters 790–793
  configuring 789–796
  daemon 799
  dpkg-reconfigure utility 789, 796
  firewall setup 784
  .forward file 796
  forwarding 794–796
  init script 784
  JumpStart: configuring **postfix** to use Gmail as a smarthost 787
  logs 590, 785, 786
  lookup tables 793
  mail directory 794
  mail.log file 786
  mailname file 787, 789
  main.cf file 787, 789
  master.cf file 794
  parameters, configuration 790–793
  port 25 782
  postmap utility 788, 793, 794
  prerequisites 784
  sasl_passwd file 788, 794
  sendmail 780
  sendmail compatibility interface 782
  smarthost 782
  testing 785
  testing with telnet 786
  tls_policy file 788, 793
postinst script (**dpkg**) 525
postmap utility 788, 793, 794
postmaster 590
postrm script (**dpkg**) 525
PostScript Printer Definition *see* PPD
power management 320
power supply *1266*
power, turning off 443
poweroff utility 441
PowerPC processor architecture 33
ppd directory (CUPS) 553
PPD files 552
PPID *see* parent process
PPP (Point-to-Point Protocol) *298*
Preboot Execution Environment 846
preinst script (**dpkg**) 525
prerm script (**dpkg**) 525
Pretty Good Privacy *see* PGP, GPG
PREVLEVEL variable 433
primary buffer *107*
printable character *1266*
printcap file 244
printenv utility 1056
printer
  see also CUPS; printing
  accepting/rejecting jobs 544t
  capability database 487
  classes 551
  configure using lpadmin 552–555
  configure using system-config-printer 542–548
  control using lpq/lpr/lprm 243–245
  default 542
  disable using cupdisable 555
  disable using cupsreject 555
  enable using cupsaccept 555
  enable using cupsenable 555
  enable/disable 544t
  IPP protocol 540
  JumpStart: setting up a remote printer 544
  network 540
  page breaks 276
  PDF, virtual 542
  print files using lpr 243–245
  print queue *540*
  printcap file 487
  queue, managing 544, 555
  queues and 540
  remote, configuring 544–548
  server settings 543
  sharing 555
  status 544t
PRINTER variable 244
printing
  see also CUPS; printer
  command line, from the 557
  command-line utilities 557t
  lpd daemon 540
  LPR line printer system 540
  quotas 554
  system *540*
  UNIX traditional 557
  Windows, from using CUPS 558
  Windows, from using Samba 559
  Windows, to using CUPS 560
private address space 667, 667t, *1266*
private key 629
privilege, least *597*, *1256*
privilege, process *599*
privileged ports *318*, *1266*
privileged user *see* **root** privileges
privileges, **root** *see* **root** privileges
problem solving 1172
/proc directory 198, 490, 498

procedural language *1086*
procedure *1266*
process *158*, *379*, 379–382, *1266*
  background 381, *1235*
  child 380, *1239*
  die 381
  files held open by a, locate using lsof 589
  first 439
  foreground *1248*
  fork *379*
  ID *see* PID
  identification *see* PID
  init 439
  kill 455–457, 501
  kill using kill 131, 172, 1072
  list using top 577
  locality 1054
  numbers, display using pidof 457
  parent 380, *1264*
  parent of all 439
  privilege 599
  search for processes using pgrep 456
  sleep 381
  spawn *see* process, fork
  spontaneous *380*
  structure 379
  substitution 419
  wake up 381
processor architecture 32
.profile file 336, *1266*
profile file 336, 487
profile.d directory 336, 487
program *1266, see also* builtins; the Utility index (1291)
  name of calling 1044
  terminating *see* kill builtin
programming language 1174t
Project Athena 15
PROM *1266*
prompts *1267*
  # 597
  $ 27
  bash 367–369
  login 440
  MariaDB 1138
  Python 1105
  representation 27
  shell 27
proprietary operating systems 9
protocols *296*, 296–298, *1267*
  connectionless *1241*
  connection-oriented *297*, *1241*
  datagram-oriented *298*
  DHCP 464–468
  HTTPS 992
  ICMP 311
  IP model 296
  IPP 540
  ISO model 297
  PPP *298*
  protocols file 488
  stream-based *297*
  TCP 296, 297
  TCP/IP 296
  UDP 296, 298
protocols file 940
proxy *322*, *1267*
proxy gateway 322, *1267*
proxy server 322, *1267*
.ps filename extension 188
ps utility 172, 345, 380, 456, 1050
PS1 variable 367, 368t
PS2 variable 351, 368
PS3 variable 369, 1035
PS4 variable 369, 1017
pseudographical interface *33*, 126
pseudoterminal 482
pstree utility 381
PTP (peer-to-peer) model 316
pts directory 160, 482
pub directory (FTP) 760
public key encryption *see* encryption, public key
Public Key Infrastructure *see* PKI
public keyserver 644
public_html directory 965
pushd builtin 356
PV *see* LVM, PV
pwd builtin 151, 192, 193, 356, 416
pwd utility 216
PWD variable 414
pwgen utility 144
PXE 846
Python 1104–1131, *1267*
  + in a print statement 1107
  append() method, list 1110
  bool() function 1123
  control structures 1114–1118
  dictionaries 1113–1114
  exception handling 1120
  file input/output 1119–1122
  file modes 1119t
  file object methods 1119t
  findall() method, re 1123
  floating point variable 1109
  for control structure 1117
  function 1107, 1107t
    defining 1124
    importing 1128
    Lambda 1129
  group() method, re 1124
  if control structure 1115
  if...elif...else control structure 1116
  if...else control structure 1115
  implied display 1105
  indenting logical blocks 1114
  invoking 1104
  items() method, dictionary 1114
  iterable *1112*, *1117*, *1118*
  keys() method, dictionary 1113
  Lambda function 1129
  len() function 1111
  libraries 1125
  list comprehensions 1130
  list methods 1111t
  lists 1109–1113
  logical blocks, indenting 1114
  map() function 1129
  match() method, re 1124
  MatchObject 1123
  methods 1108
  module, importing 1127
  namespace *1126*
  NumPy library 1126
  open() function 1119
  pickle 1122
  print statement 1107
  prompt 1105
  quotation marks, single and double 1106
  range() function 1118, 1118t

raw_input() function 1107
readlines() method, file 1120
regular expressions 1123
remove() method, list 1110
reverse() method, list 1110
scalar variable 1108
SciPy library 1126
search() method, re 1123
sort() method, list 1110
sorted() function 1111
standard input, reading from 1107
standard output, writing to 1107
strings 1118
type() function 1124
values() method, dictionary 1113
version 1104
while control structure 1117

Q

QEMU 690
QEMU/KVM 691–698
Qmail 783
qnx4 filesystem 498
Qt toolkit 17
question mark (**?**) in extended regular expressions 1168
quiescent *451*
quiet boot parameter 58
QUIT signal 131, 1070
quota utility 593
quotaon utility 593
quotation marks
see also quoting
around variables 360
around whitespace 360
double 1006
in aliases 399
in pathname expansion 418
in Python 1106
removal of (bash) 419
single 150
quoting *1267*
see also quotation marks
characters 150
let arguments 416
NEWLINE characters using \ 351
parentheses in regular expressions 1166
shell variables using \ 360
special characters in regular expressions 1165, 1165t
trap, arguments to 1070

R

radio button *1267*
RAID 45–46, 86, *1268*
backups, does not replace 568
fake 45
RAM *1268*
see also memory
disk 60, *1268*
installation requirements 31
swap and 42, 491
testing 72
random access memory *see* RAM
random bytes, generating 642
random file 482
random number generator 482
RANDOM variable 1019, 1093
randomart image (OpenSSH) *730, see also* fingerprint
RAS *1268*
raw devices *496*
raw mode, device *496*
Raymond, Eric 7
rc script *see* init script
rc.conf file 433, 451
RC5 encryption 629
rc*n*.d directory 435–437
rcS.conf file 451
rc-sysinit task 435
rc-sysinit.conf file 435
RDBMS 1137, *see also* MariaDB
RDF *1268*
RDN, LDAP *831*
read access permission *199*
read builtin 1025, 1063–1066
Readline completion commands 394–395
Readline Library command editor 392–398
Readline variables 396t
readonly builtin 362, 363
readonly memory *see* ROM
real UID *599*, *1268*
reboot system 442
reboot utility 441
recovery mode *450*, 450–453
from multiuser mode 442
rescue mode, versus 77
root password 451
root privileges 598
recursion, infinite (aliases) 401
Red Hat OpenShift 703
redirect
output of sudo using **>** 606
output of sudo using tee 606
redirection *14*, *161*, *1268, see also* standard error; standard input; standard output
redirection operators (bash) 342t
redundant array of inexpensive disks *see* RAID
reentrant code *1268*
regular character *1269*
regular expression *1161*, 1161–1168, *1269*
^ 1164
\\(...\\) brackets expressions 1166
$ 1164
ampersand in replacement strings 1167
anchors 1164
asterisks 1164, 1164t
brackets 1163, 1163t, 1166
carets and dollar signs 1164t
character class *1238*
character classes and bracketed 1169t
characters *1162*
delimiters 1162
empty 1166
extended 1167, 1168t, 1169t
pipe symbol (**|**) 1168
plus sign (**+**) 1168
question mark (**?**) 1168
summary 1169
full 1167, 1168t
list operator *see* character, class
longest match 1165
periods 1163, 1163t

regular expression, *continued*
  Python 1123
  quoted digits 1167
  quoted parentheses 1166
  quoting special characters 1165, 1165t
  replacement strings 1166, 1170t
  rules 1165
  simple strings 1162, 1162t
  special characters *1162*, 1162, 1165, 1169t
  summary 1169
reiserfs filesystem 498
relational database management system *1137, see also* MariaDB
relational operators 1079
relative pathnames 157, *190*, *1269*
releases, Ubuntu 35
religious statue, miniature *see* icon
reload utility 430
remainder operators 1082, 1084
remapping mouse buttons 116, 475
remote
  computing and local displays 472
  filesystem *1269*
  login 128
  procedure call *see* RPC
remount filesystem 207
replacement strings in regular expressions 1166
REPLY variable 1035, 1064
.repo filename extension 1182
reports, system 576, 591
repositories *122*, *511*, *515*, 515–517
repquota utility 593
request for comments *see* RFC
rescue boot parameter 77
rescue mode
  see also single-user mode
  Netboot Image, from 77
  Server Image, from 77
  versus recovery mode 77, 453
reserved ports *see* privileged ports
reset utility 458
resolution boot parameter 77
resolv.conf file 488, 894, 905, 906
resolvconf utility 488, 905
RESOLVCONF variable (DNS) 905
resolver *318*, 488, 894, *1269*
Resource Description Framework *1268*
resource records, DNS 898–900
restart utility 430
restore *1269*
restore utility 572
restricted deletion flag *see* sticky bit
restricted software package category 516
return code *see* exit status
RETURN key 27, 150, 156
reverse name resolution, DNS 902–903
RFC *1269*
rhost authentication, OpenSSH 715
.rhosts file 308
right-click, mouse 106
right-handed mouse 475
Ritchie, Dennis 10
rm utility 133, 216, 231, 493
rmdir utility 194
rmmod utility 444
rngd daemon 642
roam *1269*
role alias (Apache) 964
ROM *1270*
romfs filesystem 498
root
  see also **root** account; **root** privileges
  directory (/) *40*, 42, *185*, 189, 197, 617, *1270*
  domain (DNS) *893*, 895
  filesystem (/) *1270*
  node (DNS) *893*
  window *1270*
root account
  see also **root** privileges
  locked 98, *598*
  locking 613
  login *1270*
  password 98
  password and recovery mode 451
  password and rescue mode 451
  password and sudo 612
  prompt (#) *597*
  Ubuntu 600
  unlocking 613
/root directory 198
root privileges 98, *596–613*
  see also **root** account
  edit a file using 605, 607
  explained *596*
  gain using pkexec 604
  gaining *597–599*
  graphical programs and *597*, 604
  PATH and security 602
  prompt (#) *597*
  setuid *see* setuid
  shell with 600, 605
  sudo group and 604, 609
  using su to gain 600–602
  using sudo to gain 602–613
  wheel group and 625
root shell 600
root user *see* **root** account
rootkit *1270*
ROT13 *267*
rotate files *1270*
rotatelogs file 959
round robin *1270*
router 663, *1270*
  network *292*, 293
  set up using iptables *945–948*
row *1137*
RPC 323, *1270*
rpc file 489
rpc. prefix (daemon) 319
rpc.gssd daemon 848
rpc.idmapd daemon 848
rpc.lockd daemon 848
rpc.statd daemon 848
rpcbind
  daemon 323, 463–464, 847, 854
  display information about using rpcinfo 463–464
  service, starting 817, 823, 847, 854

rpcinfo utility 463–464, 820
RPM 1178
rpm directory 1178
.rpm filename extension 1178
rquotad daemon 321
RSA encryption 629, *1271*
rsync utility 724, 742–752
rsyncd daemon 744
rsyslog.conf file *585–586*, *587*
rsyslogd daemon *585–586*
run *1271*
Run a Command window 102, 106
run command scripts 435
/run directory 198
runlevel 438, 438t, *1271*
  DEFAULT_RUNLEVEL variable 435, 439
  emulation in Upstart 429
  event 433
  initdefault, and 435
  PREVLEVEL variable 433
  RUNLEVEL variable 433
runlevel utility 438
RUNLEVEL variable 433
run-parts utility *574*

S

Saas 703
safedit shell script 1032
salt, hash function 634
Samba *870*, *871*, *1271*
  see also Samba parameters
  Anonymous user 872
  CIFS *871*
  configure by editing **smb.conf** 880–887
  configure using shares-admin 879–880
  credentials file 876
  directory, shared 673
  firewall setup, client 874
  firewall setup, server 878
  guest user 872
  home directories, sharing 881
  [**homes**] share 881
  JumpStart: configuring a Samba server using shares-admin 879
  Linux shares, working with from Windows 877
  mapping a share *871*
  more information 871
  mount utility 876
  NBT *1261*
  net use utility (Windows) 888
  net utility 870
  net view utility (Windows) 888
  NetBIOS *1261*
  nmbd daemon 870, 888
  nmblookup utility 888
  parameters *see* Samba parameters
  password 873
  passwords 872, 873
  passwords, edit using smbpasswd 873
  passwords, Linux 872
  pdbedit utility 873
  ping utility 887
  prerequisites 878
  printing from Windows *559*
  share *871*, *1272*
  shares-admin utility 879–880
  SMB *1272*
  smb.conf file 880–887
  smbclient utility 875, 889
  smbd daemon 870
  smbpasswd file 873
  smbpasswd utility 873
  smbstatus utility 871
  smbtar utility 871
  smbtree utility 874
  testparm utility 887
  troubleshooting 887
  user map 872, *872*
  users 871
  users, list using pdbedit 873
  utilities 870t
  Windows shares 874–877, *1272*
  Windows user 871
  WINS *1281*
Samba parameters
  communication 886
  domain master browser 885
  global 882
  hosts allow 883
  hosts deny 883
  logging 885
  passdb backend 873
  security 882
  share 886
samhain utility 658
sample-spam.txt file 799
SAN *1271*
sandbox, VM *688*
sasl_passwd file 788. 794
/sbin directory 198
sbin directory 199
schema *1271*
schema directory 831
Schneier, Bruce 658
scp utility 717, 723–725, *see also* OpenSSH
screen *159*
  as standard output 160
  number, X Window System 474
script utility 265
scripts, shell *see* shell scripts
scroll *1271*
scrollbar *1271*
sdiff utility 235
sd*n* device file 481
seahorse utility 641
search
  engines *326*
  for a pattern using grep 240–242
  for files using locate 264
  for inodes using find 1016
  for keywords using apropos 137
  for open files using lsof 589
  for processes using pgrep 456
  for setgid files using find 614
  for setuid files using find 614
  for software package containing a file 514
  for software package containing a file using dpkg 530
  for software package containing a file using yum 1179
  for strings using grep 240–242
  for utilities using whereis 263
  for utilities using which 263
  path 263

Search screen *101*
Search text box 101
secret key encryption *see* encryption, symmetric key
secure file 590
.secure filename extension 638
Secure Sockets Layer *see* SSL
securenets file 826
security
  see also firewall
  access permissions 199–211
  ACL *1232*
  AIDE utility 614
  Apache directives 981
  authentication *1234*
  back door *1234*
  BIOS 32
  checksum *1239*
  chroot jail *see* chroot jail
  cipher *627*, *1239*
  ciphertext *1240*
  cleartext *1240*
  console *597*
  cookie *1241*
  cracker *1242*
  cryptography *1242 see also* cryptography
  cypher *1239*
  DDoS attack *1243*
  DNS 893
  DoS attack *1245*
  encryption *see* encryption
  FTP 754, 760, 765, 770
  GPG *see* GPG
  hole *535*
  host-based trust 308
  hping utility 657
  IP spoofing *1254*
  John the Ripper utility 657
  Kerberos *1255*
  Linux features 13
  login shell 487
  man-in-the-middle attack *631*, *1258*
  MD5 encryption *1258*
  messages 492
  MITM *see* security, man-in-the-middle attack
  nessus utility 658
  NFS 847, 855, 860
  NIS 826
  nmap utility 658
  OpenSSH 714
  PAM 440
  passphrase *627*
  password 143, 486, 615, *1264*
  PATH and **root** privileges 602
  PATH variable 366, 613
  resources 656
  .rhosts file 308
  root password 615
  RSA encryption *1271*
  samhain utility 658
  Schneier, Bruce 658
  server, securing a 616–621
  setgid files 204
  setuid files 204, 205, 614
  SHA1 hash algorithm *1272*
  SHA2 hash algorithm *1272*
  snort utility 658
  software, keeping up-to-date *511*, *535*
  spoofing *1254*
  ssh *see* ssh
  sudo group 615
  swatch utility 659
  system 615
  TCP wrappers 616–617
  telnet utility 309
  tripwire utility 659
  Trojan horse *613*, *1278*
  trusted hosts 308
  virus *1280*
  web of trust 641
  wiping a file 482
  wireshark utility 659
  worm *1282*
  xhost 473
security file 621
sed utility 592
seed, BitTorrent *531*
segment, network *291*, *1262*
select control structure 1034–1036
selection buffer *107*
self-signed certificate 630
self-signed certificates 993–994
sendmail 780, 783
  daemon 322
  masquerade 1258
  postfix compatibility interface 782
seq utility 413, 1019
sequence expression, bash 412, 1018
serial ports, setting information 458
Server edition 37
Server Image 71
Server Message Block protocol *see* Samba, SMB
servers *1271*
  see also specific server name (e.g., DNS, FTP)
  debug using telnet 310
  mail list 307
  name 316, 318
  proxy 322
  securing 616–621
  setting up 460–464, 672
  X 471, *1282*
service
  see also daemons
  configuring 436–437
  directory *1244*
  network 319
  Upstart *429*
service utility 430
services file 319, 489, 941
session *1271*
  failsafe *1247*
  initialize 440
  key, OpenSSH 631
  record using script 265
set builtin 406, 1013, 1016, 1046, 1058
set group ID *see* setgid permissions
set user id *see* setuid
set utility 1015
setfacl utility 207–211
setgid files, search for using find 614
setgid permissions 204, *1271*
setlocale() system call 375
setserial utility 458

setuid 204, *598*, *1271*
 files, dangerous *599*
 files, search for using find 614
 files, security *598*, 614
 NFS 848
 nosuid option to mount 501, 848
sexillion *1272*
sftp utility 725
.sh filename extension 336
sh Shell 334, *1236*
SHA1 hash algorithm *1272*
sha1sum utility 51, 633
SHA1SUMS file 51, 633
SHA2 hash algorithm *1272*
sha224sum utility 632, 633
sha256sum utility 633
SHA256SUMS file 633
sha384sum utility 633
sha512sum utility 633
shadow file 440, 489, 613, 636
shar file 1037
share *855*, *1272*
share directory 199
share, Samba *871*, *879*
shared memory 483
shared network topology *1272*
shares-admin utility 855–856, 858, 879–880
shebang *344*
shell 13–15, *1272*
 see also bash; bash parameters; bash variables; command line; job control; shell features; shell scripts; usage messages
 ~ (tilde) expansion 190
 archive file 1037
 Bourne (original) *1236*
 command interpreter *128*
 csh *1237*
 dash 13, *335*
 Debian Almquist *see* shell, dash
 default, change using chsh 335
 features 405–406
 filename generation 173–177
 functions *1272, see also* bash, functions
 identifying 129
 job control *see* job, control
 ksh *335*
 login *336*, *1257*
 OpenSSH 720
 options 405–406
 parameters *358*, 358–371
 pathname expansion 173–177
 prompt 27
 quoting special characters 360
 root privileges *see* **root** privileges
 sh 334, 1236
 sleep *158*
 subshell *350*
 variables *see* bash variables
shell scripts *342*, 342–346, 1003–1094, *1272*
 see also bash, functions; usage messages
 addbanner 1072
 arguments, testing 1015
 bash 1086–1096
 birthday 1036
 bundle 1037
 chkargs 1006, 1008
 chmod, using to make executable 343–344
 command_menu 1030
 comments, begin using **#** 346, 1014
 configure 534
 count 1022
 count_down 1062
 cpdir 351
 debug using **–e** and **–u** 345
 debug using **–x** 1016
 debug using **xtrace** 1088
 executing 346
 exit status 416, 1006, 1008, 1051, *1247*
 filename, unique 1050
 Here document 1036–1038
 input, read using read 1063–1066
 is_ordfile 1007
 lnks 1013
 locale 375
 locktty 1026
 makepath function 1088
 menu 1030
 out 1011
 positional parameters *1044*, 1044–1049
 quiz 1094
 recursion 1087
 safedit 1032
 shebang *344*
 shell, specify using **#!** 344, 1014
 sortmerg 1040
 special parameters 1049–1052
 spell_check 1023
 temporary files 1034, 1050
 whos 1020
 writing a basic 134
shells file 454
shift builtin 1011, 1047
shm file 483
shopt builtin 406
short-circuiting operators *349*
shortcut *see* link
shutdown utility 433, 441
signals *1069*, 1069t, *1272*
 see also signal name (e.g., KILL)
 display list of using kill 1073
Silicon Graphics *325*
simple command *152*, *155*
simple filenames 157, *189*, *1272*
single quotation marks *see* quotation marks
single-user mode 450–453
single-user system *1272*
sites-available directory 955, 958, 960
sites-enabled directory 958, 960
skel directory 566
slapcat utility 835
slapd daemon 833
sleep, shell *158*
sleep utility 1051
sleep() system call 381
slice *see* partition
slider *1272*
slow systems, installation on 32
SMART disk monitoring 91
smarthost *782*
SMB *see* Samba, SMB
smb.conf file 880–887
smbclient utility 875, 889
smbd daemon 870

smbfs filesystem 498
smbpasswd file 873
smbpasswd utility 873
smbstatus utility 871
smbtar utility 871
smbtree utility 874
SMF 427
smiley (smilies, *plural*) *1273*
SMTP 780, *1273*
SMTP relay *see* smarthost
snakeoil certificate 630, *637*
snap, window *1273*
snapshot, VM *689*
sneakernet *1273*
sniff *1273*
SNMP 681
snmpd
  daemon 681
  firewall setup 681
  snmpd file 681
  snmpd.conf file 681
snmpd.conf file 681
snort utility 658
SOA records, DNS 899
sockets *495*
SOCKS *1273*
soft links *see* links, symbolic
software
  bug *535*
  bug tracking 511
  GNU Configure and Build System *534*
  keeping up-to-date 511, *535*
  library *see* library
  termination signal 1070
  update preferences 122
  Updater window 123
  updates 123
  updating 123
software packages *510*
  adding/removing 121–125
  categories 122, 516
  contents of 524
  dependencies *510*, 512, 520, *522*
  display information about using apt-cache 522–523
  display information using dpkg *529*
  file, search for the package containing using dpkg 530
  files, list using dpkg 530
  find using yum 1179
  finding 514
  information about 526
  install using apt-get 512
  install using dpkg 527
  install/remove 123–124
  install/remove using apt-get 512–514
  install/remove using yum 1178–1179
  installing from a CD/DVD 122
  metapackage *519*
  package group *519*
  package management system, *510*
  PMS *510*
  remove configuration files using apt-get 513
  remove using apt-get 513
  remove using dpkg 528
  remove using yum 1178
  repositories *511*
  source code *525*
  source code, download using apt-get 523
  Ubuntu Software Center window 124
  update list of available using dpkg 526
  update using yum 1180
  virtual *519*
  yum repositories 1182
software-properties-gtk utility 518
sort *1273*
sort utility 168, 247–249, 419, 1040
sortmerg shell script 1040
source builtin 338
source code, download using apt-get 523
source code, **dpkg** files *525*
sources.list file *516*
sources.list.d directory 516
SPACE *1273, see also* whitespace
SPACE bar 150
spam *1273*
  see also SpamAssassin
  sample file 799
SpamAssassin 797–800
  client, using with a 800
  configuring 799
  local.cf file 799
  prerequisites 797
  running on a mail server using **postfix** 799
  sample file 799
  spam file, sample 799
  spamassassin utility 797
  spamassissin init script 798
  spamc utility 797
  spamd daemon 797, 800
  spamd init script 798
  testing 798
  user_prefs file 799
spamassassin utility 797
spamassissin init script 798
spamc utility 797
spamd daemon 797, 800
spamd init script 798
sparse file *1274*
spawn *see* fork
special characters 150, *173*, *1162*, *1274*
  ? 173
  [] 176
  * 174
  bash 372t
  filename generation 173–177
  pathname expansion 173–177
  quoting 360
  regular expressions *1161*, 1169t
special files *494*, *1239, see also* device files
special parameters 1049–1052
spell_check shell script 1023
spin box *1274*
spinner *see* spin box
splash boot parameter 58
spontaneous process *380*
spoofing, IP *1254*
spool *1274*

spool directory 199, 1172
SQL *1137*, *1274, see also* MariaDB
square brackets *1274*
square brackets, using in place of test 1008
SquirrelMail 802–804
squirrelmail-configure utility 803
sr0 file 481
src directory 199
.ssh directory 715
ssh directory 715
ssh init script 728
ssh utility 717, 720–723, *see also* OpenSSH
ssh_config file 726
ssh_host_dsa_key file 715
ssh_host_dsa_key.pub file 715
ssh_host_ecdsa_key file 715
ssh_host_ecdsa_key.pub file 715
ssh_host_ed25519_key file 715
ssh_host_ed25519_key.pub file 715
ssh_host_rsa_key file 715
ssh_host_rsa_key.pub file 715
ssh_known_hosts file 715, 719
ssh-add utility 732
ssh-agent utility 731–732
ssh-copy-id utility 730
sshd daemon 727–734
sshd_config file 730, 732
ssh-import-id utility 715
ssh-keygen utility 729–731
SSL 637
  see also SSL certificate
  Apache 992–994
  OpenSSL 637–641
SSL certificate 637–641
  generating a key pair 638
  generating a self-signed 638
  self-signed 630
  signing an 639
  snakeoil 630, *637*
stack, directory *355*, 355–357, 414
stack, PAM *621*
Stale NFS file handle error message 862
Stallman, Richard 3
standard error *159*, *339*, *1274*
  duplicate file descriptor using **1>&2** 341, 1008
  error message, sending to 1060
  file descriptor 340, 1038
  redirect 339–342
  redirect error messages to 341, 1008
  redirect using **2>** 340
  redirect using exec 1068
  redirect while redirecting standard output 340
  redirect with standard output using **&>** 171
  test if it is going to the screen 1042
standard input *159*, *1274*
  file descriptor 340, 1038
  keyboard as 160
  redirect using **<** 163–164
  redirect using **0<** 340
  redirect using exec 1068
  test if it is coming from the keyboard 1042
standard output *159*, *1274*
  duplicate file descriptor using **2>&1** 341
  file descriptor 340, 1038
  redirect and append using **>>** 165
  redirect output of sudo using tee 606
  redirect using **>** 162–163
  redirect using **1>** 340
  redirect using exec 1068
  redirect using tee 170
  redirect while redirecting standard error 340
  redirect with standard error using **&>** 171
  screen as 160
  test if it is going to the screen 1042
standards
  directories and files 197–199
  FHS (Linux Filesystem Hierarchy Standard) 197
  FSG (Free Standards Group) 197
  FSSTND (Linux Filesystem Standard) 197
  Linux 8
  LSB (Linux Standard Base) 197
  OpenPGP Message Format 641
  option handling 1076
  POSIX 8, 335
start utility 430
STARTTLS 792
startup files *188*, 335–339, *1274*
  bash 335–339
  BASH_ENV variable 337
  .bash_login 336
  .bash_logout 336
  .bash_profile 336–338, 403, 480
  bash.bashrc 484
  .bashrc 337–338, 480
  bashrc 337
  .cshrc 1242
  ENV variable 337
  .inputrc 396
  .login *1257*
  .logout *1257*
  .netrc 760
  .profile 336, *1266*
  profile 336, 487
  profile.d 487
  .toprc 578
startx utility 472
stat utility 458
statd daemon 321, 848
static IP address 304, 467, 670
status file 524
status line *1274*
status utility 430, 431
status, exit *1247*
sticky bit 204, *1275*
stop utility 430
stopping a program *see* kill builtin
stream-based protocols *297*
streaming tape *1275*
streams *see* connection-oriented protocol
strings *1275*
  comparing 1080
  length (**${...}**) 1081
  null 1263
  operators 1080t
  pattern matching 1080
  search for using grep 240–242
  within double quotation marks 360

Stroustrup, Bjarne 11
strtok() system call 996
Structured Query Language *see* SQL; MariaDB
stty utility 130, 480
Studio desktop 38
stylesheet *see* CSS
su utility 600–602
  see also **root** privileges
  root shell, spawning 600
subdirectories *185*, *1275*
subdomain (DNS) 894
subnet *304*, *1275*
  see also network, address
  address *1275*
  mask *304*, *1275*
  number *1275*
  specifying 462, 462t
subpixel hinting *1275*
subroutine *see* procedure
subshell *350*, *1275*
subtraction operator 1082
sudo group 596, 604, 609
sudo utility 98, 602–613
  see also **root** privileges
  configuring 607–613
  defaults (options) 611
  edit **sudoers** file using visudo 607
  editing a file using **–e** or sudoedit 607
  environment 606
  options 607
  redirecting output 606
  redirecting output using tee 606
  root account password and 612
  root shell, spawning 605
  sudo group 596, 604, 609
  SUDO_EDITOR variable 607
  sudoers file 607–613
  timestamps 604
SUDO_EDITOR variable 607
sudoers file 607–613
Sun Microsystems 814, 844
superblock *1275*
Superuser *98*, *596*, *1276*, *see also* **root** account; **root** privileges
suspend key (CONTROL-Z) 130, 131, 172
SVID *see* System V Interface Definition
swap *1276*
  filesystem 42, 491
  RAM and 42
  space 491, *1276*
  space, display usage using free 261
  swapon utility 491
swarm, BitTorrent *531*
swatch utility 659
switch, network 289, *291*, 662, *1262*
symbolic hostname 318
symbolic links *see* links, symbolic
symlinks *see* links, symbolic
symmetric key encryption *see* encryption, symmetric key
/sys directory 198, 492, 495
sysctl.conf file 531
syslog file 587, 590, 919
system
  see also system calls
  administrator *see* administrator
  boot failure 439
  booting 438
  characteristics of a well-maintained 426
  clock and hardware clock 226
  clock, display using date 226
  clock, set using timedatectl 579
  console *see* console
  crash *443*
  dataless *846*, *1243*
  diskless *845*
  hangs because of lack of entropy 482, 642
  logging in 99
  logs 585–586
  memory *see* memory; RAM
  messages 492
  messages, **rsyslogd** daemon 585–586
  minimal 74
  mode *1276*
  name *see* hostname
  powering down 443
  RAM *see* RAM
  rebooting 442
  reports 576, 591
  security 615
  shutting down 441
  single-user *1272*
  slow 588
system calls *11*
  exec() 380
  fork() 379, 380, 381, 996
  gethostbyname() 903
  isatty() 160
  setlocale() 375
  sleep() 381
  strtok() 996
System Settings window 113
System V *1276*
  init daemon 427
  init script *see* init script
  Interface Definition 8
system-config-printer utility 542–548
systemd daemon 427
sysv filesystem 498
SysVinit 427
SysVinit scripts *see* init script
sysv-rc-conf utility 436–437

T

T-1 line 289
T-3 line 289
TAB key 150
table, database *1137*
table, hash *1250*
TABs
  see also whitespace
  display using cat 225
tail utility 249
talk utility 322
talkd daemon 322
Tanenbaum, Andrew 5, 498
tape archive *see* tar utility
tape, streaming *1275*
tar file *257*
tar utility 257–260, 351, 569, 571t, 572
.tar.bz2 filename extension 257
.tar.gz filename extension 188, 257
.tar.Z filename extension 257

tarball *257*
task, Upstart *428*
.tbz filename extension 257
TC Shell *1276*
TCP 296, 297, *1276*
TCP wrappers 616–617, 776
TCP/IP 296
tcpdump utility 659
tcsh *1276*
tee utility 170, 606
teletypewriter *1278*
telinit utility 433, 438, 442
telnet utility 309–311, 786, 991
temporary file 1034, 1050
tera- *1276*
TERM signal 130, 131, 1070
TERM variable 129, 1176
Termcap 1175
termcap file *1277*
terminal *1277*
  ASCII *1233*
  character-based *1238*
  device 1068
  emulator *126*, 128
  emulator, open using Nautilus 110
  files 160
  interrupt signal 1069
  names 1175
  pseudo 482
  reset using reset 458
  specifying 1175
  standard input 160
  standard output 160
  virtual *see* console, virtual
  X *1282*
Terminal Server Project, Linux 845
Terminal *see* gnome-terminal
terminating execution *see* kill builtin
Terminfo 1175
terminfo file *1277*
terminology
  Apache 953
  bash features and options 405
  check box 26
  desktop 105
  DNS 905
  filesystem naming 41
  frame *26*
  installation image 36
  installation medium 36
  MariaDB 1137
  mask, subnet mask, network mask 304
  NIS 816
  partition name 41
  root privileges 599
  screen *159*
  single-user versus recovery modes 451
  SQL 1137
  switching 290
  tick 26
  Upstart daemon 428
ternary operator 1084
test builtin 1005–1007, 1007t, 1008, 1012, 1015, 1018, 1022, 1027
test utility 1042–1043
testdisk utility 231
testparm utility 887
text box *1277*
textual
  installer 78–82
  partitioning, manual 82–87
textual interface *33*
  see also command line
  advantages of 125
tftp utility 846
tftpd daemon 322
.tgz filename extension 188, 257
theme *1277*
thicknet 291, *1277*
thinnet 291, *1277*
Thompson, Ken 10, 1234
thread safe *see* reentrant code
threads in Apache 996
three-finger salute 442
thumb *1277*
Thunderbird 800
Thunderbird LDAP client 838
tick *26*, *1277*
tick box *see* check box
.tif filename extension 188, 1277
.tiff filename extension 188, 1277
tilde expansion 190, 365, 413
tildes in directory stack manipulation 414
tiled windows *1277*
time
  display using date 226
  localtime file 379
  series data *674*
  set using Date & Time window 117
  set using timedatectl *579*
  setting the system to UTC 377
  synchronize using **ntpd** 321
  timezone file 378
  to live *see* TTL
  TZ variable 377
  tzconfig utility 378
  tzselect utility 378
  zone, graphical installation 65
  zoneinfo directory 377, 378
  zones 377
time utility 255
timed daemon 322
timedatectl utility 579
timezone file 378
titlebar *106*
tls_policy file 788, 793
/tmp directory 198, 1034
toggle *1277*
token ring network *1278*
tokens *153*, 410, *1277*
toolbar *107*
tooltip *1278*
top utility *577*, 578t
topologies *288*
.toprc file 578
.torrent filename extension 531
torrent, BitTorrent *531*
Torvalds, Linus 2, 4, 7, 1255
touch utility 134, 164, 195, 251, 442, 499
tput builtin 1026
tr utility 167, 234, 266, 268, 340
traceroute utility 312
traceroute6 utility 312, 313
tracker, BitTorrent *531*
transaction signatures, DNS *see* DNS, TSIG
transient window *1278*

Transmission Control Protocol *see* TCP
Transmission Control Protocol/Internet Protocol *see* TCP/IP
transmission-cli utility *532*
transmission-show utility *533*
Transport Layer Security *see* TLS
transport layer, IP model protocol *297*
trap builtin 1026, 1069–1072
trap door *see* back door
trash, emptying 109
tripwire utility 659
Trojan horse *613*, *1278*
Trolltech 17
troubleshooting, when the system will not boot 439
true utility 349, 1071
trusted hosts 308
tset utility 458
TSTP signal 1070
TTL *312*, 900, *1278*
tty file 1068
tty utility 482
TTY *see* teletypewriter
tty1 file 434
TUI 33
tune2fs utility *504–505*
tunneling *1278*
tunneling using OpenSSH 735–737
tutorial
  ftp 757–759
  GPG 641–656
  nano 277–280
  vim 270–277
twisted pair cable 291
.txt filename extension 188
.txz filename extension 257
type builtin 1063
typeface conventions 26
typescript file 265
typeset builtin *see* declare builtin
TZ variable 377
tzconfig utility 378
tzselect utility 378

U

ubiquity partition editor 67–70
ubiquity utility 61–71
Ubuntu
  see also graphical installation; installation; installation disk; installation image; installation medium; ubiquity
  32-bit versus 64-bit 31
  booting 56
  Canonical 35
  Chinese 38
  derivatives 37
  Desktop Guide 118
  Desktop Image *see* Desktop Image
  Destkop edition 36
  downloading 29, 47–50
  DVD *see* installation image 26
  Edbuntu 37
  editions 35
  educational derivative 37
  GNOME derivative 38
  help window 118
  installation *see* installation
  IRC channels 120
  Kubuntu derivative 37
  Kylin derivative 38
  Launchpad 657
  live session *see* live session
  LTS release 35
  Lubuntu derivative 37
  Minimal edition 37
  minimal system 74
  mirrors 49
  multimedia 38
  Mythbuntu derivative 38
  Netboot edition 37
  point release 35
  recovery mode *see* recovery mode
  releases 35
  Server edition 37
  Software Center window 124
  Studio 38
  Studio derivative 38
  ubiquity installer *see* ubiquity
  upgrade to new release 64
  upgrading 26, 64, 123
  Web site, obtaining help from 119
  Xubuntu derivative 38
ubuntu (user) *59*
UCE *see* spam
uchroot.c program 619
udev utility 494
UDP (User Datagram Protocol) 296, 298, *1278*
UEFI 32, *32*, 448
ufs filesystem 498
ufw 924–927
  enabling 924
  logs 926
  ports, opening 924
  rules, deleting 926
  services 925
  status 925
UID *1279*
  display using id 601
  effective *599*, *1246*
  effective, display using whoami 600
  passwd file, in 486
  real *599*, *1268*
  real, display using who 600
umask builtin 459
umount utility 486, 501
umsdos filesystem 498
unalias builtin 399, 401
uname utility 449, 460, 720
unary operators 1081
undecillion *1279*
unetbootin utility 52
unexport a variable 1056
unicast packet *1279*
unicode *1279*
Unity
  Dash, the 101
  desktop *see* desktop
  Favorites list 101
  Nautilus *see* Nautilus
  object *see* object
  panel *see* panel
  Run a Command window 102, 106
  Search text box 101

Systems Settings window 113
unity-tweak-tool utility 102
workspace *see* workspace
unity-control-center utility *564–566*
unity-tweak-tool utility 102
universe software package category *516*
University of Illinois 325
UNIX
Bourne Shell 334
Linux roots in 2
philosophy 307
printing, traditional *557*
System V 3, *1276*
System V Interface Definition 8
unix2dos utility 268
unix2mac utility 268
unlocking the **root** account 613
unmanaged window *1279*
unmount a filesystem using umount 501
unset builtin 362
unshielded twisted pair *see* UTP
until control structure 1025–1027
unxz utility 256
unzip utility 257
updatedb utility 264
upgrading Ubuntu 26, 64, 123
Upstart
see also Upstart daemon
DEFAULT_RUNLEVEL variable 435, 439
event 432
initctl utility 429
rc-sysinit task 435
reload utility 430
restart utility 430
start utility 430
status utility 430
stop utility 430
Upstart daemon *427–435*
see also Upstart
anacron and *575*
communicate with Upstart using initctl 429
event *428*
job definition files 433–*435*
jobs *428*, 430–432
rc task 433
rc-default task 435
runlevel 438
runlevel emulation 429
runlevel event 433
runlevel utility 438
service *429*
shutdown utility 433
starting 439
status utility 431
task *428*
telinit utility 433, 438
terminology 428
tty*n* tasks 434
uptime utility 261
uptime, display using w 262
urandom file 482
URI *1279*
URL *326*, *1279*
usage messages *152*, *1008*, 1011, 1014, 1023, *1279, see also* error messages; messages
USB devices, list using lsusb 666
user
see also user accounts; usernames
ID *see* UID
interface *1279*
map, Samba 872
mode *1279*
name *see* usernames
private groups 485
Superuser *see* **root** account
ubuntu *59*
user accounts
see also user; usernames; **root** account
add using User Accounts window 118
add using useradd 566
change using chage 567
graphical installation 66
manage using unity-control-center *564–566*
modify using User Accounts window 118
modify using usermod *567*
remove using userdel 566
User Datagram Protocol *see* UDP
user_prefs file 799
useradd utility *566*
userdel utility *566*
usermod utility *567*, 567, 604
usernames *1279*
display using w 262
in the **passwd** file 486
list using who 262
root *see* **root** account
userspace *1280*
/usr directory 198
UTC 377, *1280*
UTF-8 1280
utilities 18, *1280, see also* commands; the Utility index (1291); inside front and back covers
alternative names 483
backup 569
basic 132–134
builtin *178*
builtins versus 1006
links to 483
locate using whereis 263
locate using which 263
names, typeface 27
UTP cable 291
UUID *1280*
fstab, in 502
number, display using blkid 502
numbers, device 481

V

/var directory 43, 199
.var filename extension 966, 985
variables *1280*
see also bash variables
completion 395
environment 1054
Python 1108
shell 1053
version control, Bazaar 511
vfat filesystem 498
VG *see* LVM, VG
vi *see* vim
viewport *see* workspace
vim

case sensitivity *1238*
Command mode 272
correcting a mistake 276
correcting text 274
deleting text 275
exit, emergency 272
help system 273
Input mode 272, 273
inserting text 276
Last Line mode 272
moving the cursor 274
Normal mode *see* vim, Command mode
page breaks 276
quitting 276
safedit script 1032
starting 271
terminal, specifying 1175
undoing changes 275
vimtutor utility 270
Work buffer 276
virsh utility 697
virt-manager utility 690, 693–697
virtual
see also *VM*
console 61, *127*, *1280*
machine *see* VM
memory and swap space 491
memory, report on using vmstat *576*
package *519*
private network *see* VPN
software packages *519*
termianl *see* virtual, console
VirtualBox 690
virt-viewer utility 696
viruses *1280*
VISUAL variable 607
visudo utility 607
VLAN *1280*
VM 688–703
see also virtual
advantages 688
CPU extensions 692
disadvantages 689
domain *688*
ESXi 690
GNOME Boxes 690
gnome-boxes utility 690
guest *688*
host *688*
host resources 692
hypervisor *688*
KVM 690
libvirt 690
libvirtd group 692, 698
node *688*
package, **virtualization** 692
QEMU 690
QEMU/KVM 691–698
sandbox *688*
snapshot *689*
virsh utility 697
virt-manager utility 693–697
VirtualBox 690
virt-viewer utility 696
vmstat utility *576*
VMware 690
VMware ESXi 690
VMware Player 698–703
VMware Tools 702
Xen 690
vmstat utility *576*
VMware 690
VMware ESXi 690
VMware Player 698–703
VMware Tools 702
volume label 455
VPN *287*, *1280*
vsftpd
see also FTP; FTP clients
configuration files 766, 777
configuration parameters
connection 774
display 773
download 770
log 774
logging in 768
message 772
miscellaneous 776
server setup 767
upload 770
daemon 764
firewall setup 765
ftp directory 769
ftp file 765, 769
init script 765
logvsftpd.log file 777
more information 755
prerequisites 764
running in a chroot jail 769
setting up 764–777
testing 765
vsftpd.banned_emails file 769
vsftpd.chroot_list file 777
vsftpd.conf file 766
vsftpd.log file 774
vsftpd.pem file 776
vsftpd.user_list file 768, 777
vsftpd.banned_emails file 769
vsftpd.chroot_list file 777
vsftpd.log file 774
vsftpd.pem file 776
vsftpd.user_list file 768
vt100/vt102/vt104 terminal 1175
Vulcan death grip 442
VxFS filesystem 498

W

w utility 262
W2K *1281*
W3 *see* World Wide Web
W3C *1281*
WAN *293*, *1281*
WAP 663, *1281*
wc utility 252, 415
Web
see also World Wide Web
crawler *326*
ring *1281*
web of trust 641
webalizer utility 997
Webmail 801–804
weekly file 575
Weissman, Terry *536*
whatis utility 70, 137, 432
wheel group and su 625
whereis utility 263
which utility 263
while control structure 1021–1024
whiptail utility 1066
whitespace *150*, *152*, *1281*
on the command line 348
quoting 360
who utility 262, 438, 600
whoami utility 600

whois utility 314
whos shell script 1020
wide area network *see* WAN
Widenius, Michael "Monty" 1136
widget *1281, see also* GUI
Wi-Fi *1281, see also* wireless
wildcards *173*, *1281, see also* special characters
window *106*, *1281*
  see also screen
  cascading *1238*
  clipboard 108
  cut and paste 107
  cycling *102*
  focus, input 102
  ignored *1253*
  input focus *102*
  manager *16*, 17, *1281*
  menubar *106*
  minimize *1260*
  moving 107
  Object Properties 111–113
  Operations menu 106
  resizing 107
  root *1270*
  Run a Command 106
  scrollbar *1271*
  slider *1272*
  snap *1273*
  thumb *1277*
  tiled *1277*
  titlebar *106*
  toolbar *107*
  transient *1278*
  unmanaged *1279*
  working with 107
Windows
  see also Samba
  convert files from/to Linux format 268
  dual-boot system 91–93
  file namespace versus Linux 40
  filename limitations 187
  formatting *38*
  integration *see* Samba
  net use utility (Samba) 888
  net view utility (Samba) 888
  networks, browse using Samba 875
  print from, using CUPS 558
  print from, using Samba 559
  print to, using CUPS 560
  shares
    see also Samba, share
    mounting 876
    working with using Samba 874
WINS *1281*
WinZip 257
wiping a file 482
wire *see* cable
wireless
  802.11 specification *1232*
  access point 663, *1281*
  ad hoc mode 664
  bridge 664
  infrastructure mode 664
  network 293
wireshark utility 659
words *130*, *153*, *1282*
  erase key (CONTROL-W) 130, 156
  on the command line 410
  splitting *369*, 417
Work buffer *1282*
working directory *151*, *1282*
  change to another using cd 193
  executing a file in 344, 366
  relative pathnames and 190
  significance of 190
  versus home directory 194
workspace *105*, *1282*
workstation 9, *1282*
World Wide Web 325
  browsers *325*, 326
  Consortium *1281*
  hypertext *325*
  Mosaic browser *325*
  Netscape Navigator *325*
  search engines *326*
  URLs *326*
  Web crawler *326*
worms *1282*
write access permission *199*
wtmp file 584
WWW *see* World Wide Web
WYSIWYG *1282*

X

X Consortium 15
X server *1282*
X terminal *1282*
X utility 16
X Window System 15, 16, 471–476, *1282*
  client and server 471
  display number 474
  ––display option 475
  DISPLAY variable 474
  display, access to 473
  events *471*
  exiting from 475
  freedesktop.org group 18
  ID string 474
  –nolisten tcp option 472
  remote computing and local displays 472
  screen number 474
  server 471
  starting 472
  startx utility 472
  X terminal *1282*
  X11 forwarding, OpenSSH 717, 726, 734, 736
  xev utility 472
  XFree86 versus X.org 16
  xhost utility 473
  Xinerama *1282*
  xmodmap utility 475
X11 directory 198
x86 processor architecture 33
x86_64 processor architecture 33
x86_64-linux-gnu directory 199
xargs utility 268, 592
XDMCP *1282*
xDSL *1282*
Xen 690
xev utility 472
Xfce desktop, installing 125
XFS filesystem 498, 506–507
xfs_fsr utility 507
xfs_repair utility 506
xhost utility 473
.xhtml filename extension 966
Xinerama *1282*
XINU 5

XML *1282*
xmodmap utility 475
XSM *1283*
xterm terminal name 1175
Xubuntu 38
xz utility 253–257
xzgrep utility 256
xzless utility 256

Y

Yellow Pages 814
yp.conf file 819
ypbind daemon 818, 820
ypcat utility 816
ypinit utility 828
ypmatch utility 816
yppasswd utility 821–822
yppasswdd daemon 830
yppush utility 826
ypserv.conf file 824
ypwhich utility 820
ypxfr utility 823
ypxfrd daemon 823
yum
  commands 1181
  remove option 1178
  repositories 1182
  update option 1180
  updating packages 1180
  yum.conf file 1182
  yum.repos.d file 1182
yum.conf file 1182
yum.repos.d file 1182

Z

.Z filename extension 188, 257
Z Shell *1283*
zcat utility 256
zero file 483
zgrep utility 256
Zimmermann, Phil 641
.zip filename extension 257
zip utility 254, 257
zless utility 256
zoneinfo directory 377, 378
zones, DNS 895
zsh shell *1283*
zulu time *see* UTC