The Complete Reference

Covers Linux kernel 2.6 and applies to ALL Linux distributions

Linux

Sixth Edition

- Manage and secure Linux from the desktop, shell, or command line
- Configure the latest Internet applications and services
- Administer users, file systems, networks, and devices

Richard Petersen

McGraw Hill Osborne

www.allitebooks.com

Linux: The Complete Reference, Sixth Edition

www.allitebooks.com

About the Author

Richard Petersen, MLIS, teaches Unix and C/C++ courses at the University of California at Berkeley. He is the author of *Linux: The Complete Reference* (all six editions), *Red Hat Enterprise and Fedora Linux: The Complete Reference*, *Red Hat Linux*, *Linux Programming*, *Red Hat Linux Administrator's Reference*, *Linux Programmer's Reference*, *Introductory C with C++*, *Introductory Command Line Unix for Users*, and many other books. He is a contributor to linux.sys-con.com (*Linux World Magazine*) with articles on IPv6, the Fedora operating system, Yum, Fedora repositories, the Global File System (GFS), udev device management, and the Hardware Abstraction Layer (HAL).

About the Technical Editor

Dean Henrichsmeyer has served as technical editor for a previous edition of *Linux: The Complete Reference* and for several editions of another book, *Red Hat Linux: The Complete Reference*. He holds a B.S. in Computer Science and has been working with Linux for more than a decade. He is currently a site director for SourceForge, Inc., the media group responsible for websites such as SourceForge.net, Linux.com, Slashdot.org, freshmeat.net, and ThinkGeek.com.

www.allitebooks.com

Linux: The Complete Reference, Sixth Edition

Richard Petersen



New York Chicago San Francisco
Lisbon London Madrid Mexico City
Milan New Delhi San Juan
Seoul Singapore Sydney Toronto

www.allitebooks.com

The McGraw-Hill Companies

Copyright © 2008 by The McGraw-Hill Companies. All rights reserved. Manufactured in the United States of America. Except as permitted under the United States Copyright Act of 1976, no part of this publication may be reproduced or distributed in any form or by any means, or stored in a database or retrieval system, without the prior written permission of the publisher.

0-07-159664-X

The material in this eBook also appears in the print version of this title: 0-07-149247-X.

All trademarks are trademarks of their respective owners. Rather than put a trademark symbol after every occurrence of a trademarked name, we use names in an editorial fashion only, and to the benefit of the trademark owner, with no intention of infringement of the trademark. Where such designations appear in this book, they have been printed with initial caps.

McGraw-Hill eBooks are available at special quantity discounts to use as premiums and sales promotions, or for use in corporate training programs. For more information, please contact George Hoare, Special Sales, at george_hoare@mcgraw-hill.com or (212) 904-4069.

TERMS OF USE

This is a copyrighted work and The McGraw-Hill Companies, Inc. ("McGraw-Hill") and its licensors reserve all rights in and to the work. Use of this work is subject to these terms. Except as permitted under the Copyright Act of 1976 and the right to store and retrieve one copy of the work, you may not decompile, disassemble, reverse engineer, reproduce, modify, create derivative works based upon, transmit, distribute, disseminate, sell, publish or sublicense the work or any part of it without McGraw-Hill's prior consent. You may use the work for your own noncommercial and personal use; any other use of the work is strictly prohibited. Your right to use the work may be terminated if you fail to comply with these terms.

THE WORK IS PROVIDED "AS IS." McGRAW-HILL AND ITS LICENSORS MAKE NO GUARANTEES OR WARRANTIES AS TO THE ACCURACY, ADEQUACY OR COMPLETENESS OF OR RESULTS TO BE OBTAINED FROM USING THE WORK, INCLUDING ANY INFORMATION THAT CAN BE ACCESSED THROUGH THE WORK VIA HYPERLINK OR OTHERWISE, AND EXPRESSLY DISCLAIM ANY WARRANTY, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. McGraw-Hill and its licensors do not warrant or guarantee that the functions contained in the work will meet your requirements or that its operation will be uninterrupted or error free. Neither McGraw-Hill nor its licensors shall be liable to you or anyone else for any inaccuracy, error or omission, regardless of cause, in the work or for any damages resulting therefrom. McGraw-Hill has no responsibility for the content of any information accessed through the work. Under no circumstances shall McGraw-Hill and/or its licensors be liable for any indirect, incidental, special, punitive, consequential or similar damages that result from the use of or inability to use the work, even if any of them has been advised of the possibility of such damages. This limitation of liability shall apply to any claim or cause whatsoever whether such claim or cause arises in contract, tort or otherwise.

www.allitebooks.com

Professional



Want to learn more?

We hope you enjoy this McGraw-Hill eBook! If you'd like more information about this book, its author, or related books and websites, please click here.

www.allitebooks.com

To my nieces,
Aleina and Larisa

www.allitebooks.com

This page intentionally left blank

www.allitebooks.com

Contents at a Glance

Part I Introduction

1 Introduction to Linux 3
2 Getting Started 17

Part II The Linux Shell and File Structure

3 The Shell 35
4 The Shell Scripts and Programming 65
5 Shell Configuration 89
6 Linux Files, Directories, and Archives 115

Part III Desktop

7 The X Window System, Xorg, and Display Managers 145
8 GNOME 169
9 KDE 197

Part IV Linux Software

10 Software Management 219
11 Office and Database Applications 237
12 Graphics Tools and Multimedia 255
13 Mail and News Clients 265
14 Web, FTP, and Java Clients 281
15 Network Tools 301

Part V Security

16 Encryption, Integrity Checks, and Signatures 313
17 Security-Enhanced Linux 327
18 IPsec and Virtual Private Networks 349
19 Secure Shell and Kerberos 359
20 Firewalls 373

www.allitebooks.com

Part VI Internet and Network Services

21 Managing Services 401
22 FTP Servers 423
23 Web Servers 443
24 Proxy Servers 467
25 Mail Servers 477
26 Print, News, Search, and Database Servers 503

Part VII System Administration

27 Basic System Administration 523
28 Managing Users 551
29 File Systems 583
30 RAID and LVM 615
31 Devices and Modules 639
32 Kernel Administration 671
33 Backup Management 693

Part VIII Network Administration Services

34 Administering TCP/IP Networks 707
35 Network Autoconfiguration with IPv6, DHCPv6, and DHCP 745
36 NFS and NIS 761
37 Distributed Network File Systems 777
A Where to Obtain Linux Distributions 785

Index 787

www.allitebooks.com

For more information about this title, click here

Contents

Acknowledgments xxix
Introduction xxxi

Part I **Introduction**

1 **Introduction to Linux** **3**
Linux Distributions 4
Operating Systems and Linux 6
History of Linux and Unix 6
Unix 7
Linux 7
Linux Overview 8
Open Source Software 9
Linux Software 10
Software Repositories 10
Third-Party Linux Software Repositories 11
Linux Office and Database Software 11
Internet Servers 12
Development Resources 13
Online Linux Information Sources 13
Linux Documentation 13

2 **Getting Started** **17**
Install Issues 17
Accessing Your Linux System 19
The Display Managers: GDM and KDM 19
Switching Users 20
Accessing Linux from the Command Line Interface 20
The GNOME and KDE Desktops 22
KDE 22
XFce4 22
GNOME 22
GNOME and KDE Applets 23
Starting a GUI from the Command Line 24
Desktop Operations 24
Desktop Themes 24
Fonts 25
Configuring Your Personal Information 26
Sessions 27
Using Removable Devices and Media 27
Installing Multimedia Support: MP3, DVD, and DivX 27
Command Line Interface 27

Help Resources 28
Context-Sensitive Help 29
Application Documentation 29
The Man Pages 29
The Info Pages 29
Software Repositories 30
Windows Access and Applications 30
Setting Up Windows Network Access: Samba 30
Running Windows Software on Linux: Wine 31

Part II The Linux Shell and File Structure

3 The Shell 35
The Command Line 35
Command Line Editing 37
Command and Filename Completion 38
History 40
History Events 40
History Event Editing 42
Configuring History: HISTFILE and HISTSAVE 43
Filename Expansion: *, ?, [ ] 43
Matching Multiple Characters 45
Matching Single Characters 45
Matching a Range of Characters 46
Matching Shell Symbols 46
Generating Patterns 47
Standard Input/Output and Redirection 47
Redirecting the Standard Output: > and >> 48
The Standard Input 50
Pipes | 50
Redirecting and Piping the Standard Error: >&, 2> 51
Jobs: Background, Kills, and Interruptions 52
Running Jobs in the Background 53
Referencing Jobs 54
Job Notification 54
Bringing Jobs to the Foreground 54
Canceling Jobs 55
Suspending and Stopping Jobs 55
Ending Processes: ps and kill 55
The C Shell: Command Line Editing and History 56
C Shell Command Line Editing 57
C Shell History 57
The TCSH Shell 62
TCSH Command Line Completion 62
TCSH History Editing 62
The Z-shell 63

4 The Shell Scripts and Programming 65
Shell Variables 66
Definition and Evaluation of Variables: =, $, set, unset 66

Variable Values: Strings 67
Values from Linux Commands: Back Quotes 70
Shell Scripts: User-Defined Commands 70
Executing Scripts 71
Script Arguments 71
Environment Variables and Subshells: export and setenv 73
Shell Environment Variables 75
TCSH/C Shell Environment Variables 76
Control Structures 77
Test Operations 77
Conditional Control Structures 78
Loop Control Structures 81
TCSH/C Shell Control Structures 81
Test Expressions 82
TCSH Shell Conditions: if-then, if-then-else, switch 82
TCSH Shell Loops: while and foreach 86

5 Shell Configuration 89
Shell Initialization and Configuration Files 90
Configuration Directories and Files 90
Aliases 91
Aliasing Commands and Options 92
Aliasing Commands and Arguments 92
Aliasing Commands 93
Controlling Shell Operations 93
Environment Variables and Subshells: export 94
Configuring Your Shell with Shell Parameters 94
Shell Parameter Variables 95
Configuring Your Login Shell: .bash_profile 101
Configuring the BASH Shell: .bashrc 105
The BASH Shell Logout File: .bash_logout 106
The TCSH Shell Configuration 107
TCSH/C Aliases 107
TCSH/C Shell Feature Variables: Shell Features 108
TCSH/C Special Shell Variables for Configuring Your System 109
TCSH/C Shell Initialization Files: .login, .tcshrc, .logout 111

6 Linux Files, Directories, and Archives 115
Linux Files 116
The File Structure 117
Home Directories 118
Pathnames 118
System Directories 119
Listing, Displaying, and Printing Files: ls, cat, more, less, and lpr 119
Displaying Files: cat, less, and more 120
Printing Files: lpr, lpq, and lprm 121
Managing Directories: mkdir, rmdir, ls, cd, and pwd 121
Creating and Deleting Directories 122
Displaying Directory Contents 123
Moving Through Directories 123
Referencing the Parent Directory 124

File and Directory Operations: find, cp, mv, rm, and ln 124
Searching Directories: find 124
Copying Files 126
Moving Files 129
Copying and Moving Directories 129
Erasing Files and Directories: The rm Command 130
Links: The ln Command 130
The mtools Utilities: msdos 132
Archiving and Compressing Files 133
Archiving and Compressing Files with File Roller 133
Archive Files and Devices: tar 134
File Compression: gzip, bzip2, and zip 138

Part III Desktop

7 The X Window System, Xorg, and Display Managers 145
The X Protocol 146
Xorg 147
Xorg Configuration: /etc/X11/xorg.conf 149
Screen 150
Files, Modules, and ServerFlags 151
Input Device 152
Monitor 153
Device 154
ServerLayout 154
Multiple Monitors 155
X Window System Command Line Arguments 155
X Window System Commands and Configuration Files 156
XFS Fonts 158
X Resources 158
X Commands 160
Display Managers: XDM, GDM, and KDM 160
Xsession 162
The X Display Manager (XDM) 163
The GNOME Display Manager 164
The K Display Manager (KDM) 166
X Window System Command Line Startup: startx, xinit, and xinitrc 167

8 GNOME 169
GNOME 2.*x* Features 170
GTK+ 171
The GNOME Interface 171
GNOME Components 173
Quitting GNOME 173
GNOME Help 173
The GNOME Desktop 174
Drag and Drop Files to the Desktop 174
Applications on the Desktop 175
GNOME Desktop Menu 175
Window Manager 175

The GNOME Volume Manager 176
The GNOME File Manager: Nautilus 178
Nautilus Window 178
Nautilus Sidebar: Tree, History, and Notes 180
Displaying Files and Folders 180
Nautilus Menu 181
Navigating Directories 181
Managing Files 182
Application Launcher 184
File and Directory Properties 184
Nautilus Preferences 186
Nautilus as a FTP Browser 186
The GNOME Panel 187
Panel Properties 187
Panel Objects 189
Special Panel Objects 191
GNOME Applets 191
Workspace Switcher 192
GNOME Window List 192
GNOME Configuration 193
GNOME Directories and Files 193
GNOME User Directories 194
The GConf Configuration Editor 194

9 KDE 197
The Qt Library 198
Configuration and Administration Access with KDE 199
The KDE Desktop 199
KDE Menus 200
Quitting KDE 201
KDE Desktop Operations 201
Accessing System Resources from the File Manager 202
Configuring Your Desktop 203
Desktop Link Files and URL Locations 203
KDE Windows 204
Virtual Desktops: The KDE Desktop Pager 205
KDE Panel: Kicker 205
The KDE Help Center 206
Applications 207
Mounting Devices from the Desktop 208
KDE File Manager and Internet Client: Konqueror 208
Konqueror Window 209
Navigation Panel 210
Search 211
Navigating Directories 211
Copy, Move, Delete, Rename, and Link Operations 212
Web and FTP Access 213
Configuring Konqueror 213
KDE Configuration: KDE Control Center 214
.kde and Desktop User Directories 215

MIME Types and Associated Applications 215
KDE Directories and Files 216

Part IV Linux Software

10 Software Management 219
Software Package Types 219
Downloading ISO and DVD Distribution Images with BitTorrent 220
Red Hat Package Manager (RPM) 221
The rpm Command 222
Querying Information from RPM Packages and Installed Software 224
Installing and Updating Packages with rpm 226
Removing RPM Software Packages 226
RPM: Verifying an RPM Installation 226
Rebuilding the RPM Database 227
Debian 227
Installing Software from Compressed Archives: .tar.gz 228
Decompressing and Extracting Software in One Step 228
Decompressing Software Separately 229
Selecting an Install Directory 230
Extracting Software 230
Compiling Software 231
Configure Command Options 232
Development Libraries 232
Shared and Static Libraries 232
Makefile File 233
Command and Program Directories: PATH 233
/etc/profile 234
.bash_profile 234
Subversion and CVS 235
Packaging Your Software with RPM 235

11 Office and Database Applications 237
Running Microsoft Office on Linux: CrossOver 238
OpenOffice.org 239
KOffice 241
KOffice Applications 241
KParts 242
GNOME Office 243
Document Viewers (PostScript, PDF, and DVI) 244
PDA Access 245
Database Management Systems 245
SQL Databases (RDMS) 245
Xbase Databases 248
Editors 248
GNOME Editor: Gedit 248
K Desktop Editors: Kate, KEdit, and KJots 248
The Emacs Editor 249
The Vi Editor: Vim and Gvim 250

12 Graphics Tools and Multimedia 255
Graphics Tools 255
Photo Management Tools: F-Spot and digiKam 256
KDE Graphics Tools 257
GNOME Graphics Tools 257
X Window System Graphic Programs 257
Multimedia 258
GStreamer 259
Sound Applications 260
CD Burners and Rippers 261
Video Applications 262

13 Mail and News Clients 265
Mail Clients 265
MIME 266
Evolution 267
Thunderbird 268
GNOME Mail Clients: Evolution, Balsa, and Others 269
The K Desktop Mail Client: KMail 270
SquirrelMail Web Mail Client 270
Emacs 271
Command Line Mail Clients 271
Notifications of Received Mail 273
Accessing Mail on Remote POP Mail Servers 274
Mailing Lists 275
Usenet News 275
Newsreaders 277
News Transport Agents 278

14 Web, FTP, and Java Clients 281
Web Clients 281
URL Addresses 282
Web Browsers 282
Creating Your Own Website 286
Java for Linux 287
Sun, Java-like, JPackage, and Blackdown 287
Installing the Java Runtime Environment: JRE 289
Enabling the Java Runtime Environment for Mozilla/Firefox 289
The Java Applications 289
The Java 2 Software Development Kit 289
FTP Clients 290
Network File Transfer: FTP 290
Web Browser–Based FTP: Firefox 291
The K Desktop File Manager: Konqueror 292
GNOME Desktop FTP: Nautilus 292
gFTP 292
wget 293
curl 293
ftp 293
Automatic Login and Macros: .netrc 297

lftp 298
NcFTP 299

15 Network Tools 301
Network Information: ping, finger, traceroute, and host 301
GNOME Network Tools: gnome-nettool 301
ping 302
finger and who 303
host 303
traceroute 303
Network Talk and Messenger Clients: VoIP, ICQ, IRC, AIM, and Talk 304
Ekiga 304
ICQ 305
Instant Messenger 305
Telnet 306
RSH, Kerberos, and SSH Remote Access Commands 307
Remote Access Information 308
Remote Access Permission: .k5login 308
rlogin, slogin, rcp, scp, rsh, and ssh 309

Part V Security

16 Encryption, Integrity Checks, and Signatures 313
Public Key Encryption, Integrity Checks, and Digital Signatures 313
Public-Key Encryption 314
Digital Signatures 314
Integrity Checks 314
Combining Encryption and Signatures 315
GNU Privacy Guard 316
GnuPG Setup: gpg 318
Using GnuPG 321
Checking Software Package Digital Signatures 323
Importing Public Keys 323
Validating Public Keys 324
Checking RPM Packages 324
Intrusion Detection: Tripwire and AIDE 325
Encrypted File Systems 326

17 Security-Enhanced Linux 327
Flask Architecture 327
System Administration Access 328
Terminology 329
Identity 329
Domains 330
Types 330
Roles 330
Security Context 331
Transition: Labeling 331
Policies 331
Multi-Level Security (MLS) and Multi-Category Security (MCS) 331
Management Operations for SELinux 332

Turning Off SELinux 332
Checking Status and Statistics 332
Checking Security Context 333
SELinux Management Tools 333
semanage 334
The Security Policy Analysis Tool: apol 334
Checking SELinux Messages: seaudit 334
Allowing Access: chcon and audit2allow 334
The SELinux Reference Policy 335
Multi-Level Security (MLS) 336
Multi-Category Security (MCS) 336
Policy Methods 336
Type Enforcement 336
Role-Based Access Control 336
SELinux Users 336
Policy Files 337
SELinux Configuration 337
SELinux Policy Rules 337
Type and Role Declarations 338
File Contexts 339
User Roles 339
Access Vector Rules: allow 339
Role Allow Rules 340
Transition and Vector Rule Macros 340
Constraint Rules 340
SELinux Policy Configuration Files 340
Compiling SELinux Modules 341
Using SELinux Source Configuration 341
Interface Files 342
Types Files 343
Module Files 343
Security Context Files 343
User Configuration: Roles 343
Policy Module Tools 343
Application Configuration: appconfig 344
Creating an SELinux Policy: make and checkpolicy 344
SELinux: Administrative Operations 345
Using Security Contexts: fixfiles, setfiles, restorecon, and chcon . . . 345
Adding New Users 345
Runtime Security Contexts and Types: contexts 346

18 IPsec and Virtual Private Networks 349
IPsec Protocols 349
IPsec Modes 350
IPsec Security Databases 350
IPsec Tools 351
Configuring Connections with setkey 351
Security Associations: SA 351
Security Policy: SP 352
Receiving Hosts 352
Two-Way Transmissions 353

Configuring IPsec with racoon: IKE 354
Certificates 355
Connection Configuration with racoon 355
IPsec and IP Tables: Net Traversal 355
IPsec Tunnel Mode: Virtual Private Networks 356

19 Secure Shell and Kerberos 359
The Secure Shell: OpenSSH 359
SSH Encryption and Authentication 360
SSH Tools 361
SSH Setup 362
SSH Clients 365
Port Forwarding (Tunneling) 367
SSH Configuration 368
Kerberos 368
Kerberos Servers 369
Authentication Process 369
Kerberized Services 371
Configuring Kerberos Servers 371

20 Firewalls 373
Firewalls: IPtables, NAT, and ip6tables 373
IPtables 374
ip6tables 374
Modules 375
Packet Filtering 375
Chains 375
Targets 376
Firewall and NAT Chains 376
Adding and Changing Rules 376
IPtables Options 379
Accepting and Denying Packets: DROP and ACCEPT 379
User-Defined Chains 380
ICMP Packets 381
Controlling Port Access 382
Packet States: Connection Tracking 383
Specialized Connection Tracking: ftp, irc, Amanda, tftp 384
Network Address Translation (NAT) 384
Adding NAT Rules 384
NAT Targets and Chains 385
NAT Redirection: Transparent Proxies 386
Packet Mangling: The Mangle Table 386
IPtables Scripts 387
An IPtables Script Example: IPv4 387
IP Masquerading 395
Masquerading Local Networks 395
Masquerading NAT Rules 396
IP Forwarding 396
Masquerading Selected Hosts 396

www.allitebooks.com

Part VI Internet and Network Services

21 Managing Services 401
System Startup Files: /etc/rc.d 401
rc.sysinit and rc.local 401
/etc//init.d 402
SysV Init: init.d Scripts 403
Starting Services: Standalone and xinetd 404
Starting Services Directly 405
Starting and Stopping Services with Service Scripts 406
Starting Services Automatically 406
Service Management: chkconfig, services-admin, rrconf, sysv-rc-conf, and update-rc.d 407
chkconfig 407
rcconf, services-admin, sysv-rc-conf, and update-rc.d 410
Service Scripts: /etc/init.d 412
Service Script Functions 412
Service Script Tags 413
Service Script Example 414
Installing Service Scripts 415
Extended Internet Services Daemon (xinetd) 415
Starting and Stopping xinetd Services 416
xinetd Configuration: xinetd.conf 416
xinetd Service Configuration Files: /etc/xinetd.d Directory 417
Configuring Services: xinetd Attributes 418
Disabling and Enabling xinetd Services 418
TCP Wrappers 421

22 FTP Servers 423
FTP Servers 423
Available Servers 424
FTP Users 424
Anonymous FTP: vsftpd 425
The FTP User Account: anonymous 425
FTP Group 425
Creating New FTP Users 426
Anonymous FTP Server Directories 426
Anonymous FTP Files 427
Using FTP with rsync 427
Accessing FTP Sites with rsync 427
Configuring an rsync Server 428
rsync Mirroring 429
The Very Secure FTP Server 429
Running vsftpd 429
Configuring vsftpd 430
vsftpd Access Controls 433
vsftpd Virtual Hosts 434
vsftpd Virtual Users 435
Professional FTP Daemon: ProFTPD 436
Install and Startup 436

Authentication 436
proftpd.config and .ftpaccess 436
Anonymous Access 438
Virtual FTP Servers 440

23 Web Servers 443
Tux 443
Alternate Web Servers 444
Apache Web Server 444
Java: Apache Jakarta Project 445
Linux Apache Installations 446
Apache Multiprocessing Modules: MPM 447
Starting and Stopping the Web Server 447
Apache Configuration Files 448
Apache Configuration and Directives 448
Global Configuration 449
Server Configuration 451
Directory-Level Configuration: .htaccess and <Directory> 452
Access Control 453
URL Pathnames 453
MIME Types 454
CGI Files 455
Automatic Directory Indexing 455
Authentication 456
Log Files 457
Virtual Hosting on Apache 458
IP-Based Virtual Hosting 459
Name-Based Virtual Hosting 459
Dynamic Virtual Hosting 459
Server-Side Includes 462
PHP 463
Apache Configuration Tool 463
Web Server Security: SSL 464

24 Proxy Servers 467
Configuring Client Browsers 468
The squid.conf File 469
Security 470
Caches 473
Connecting to Caches 473
Memory and Disk Configuration 474
Administrative Settings 474
Logs 474
Web Server Acceleration: Reverse Proxy Cache 474

25 Mail Servers 477
Mail Transport Agents 477
Received Mail: MX Records 478
Postfix 479
Postfix Commands 479
Postfix Configuration: main.cf 480

Postfix Greylisting Policy Server 482
Controlling User and Host Access 483
Sendmail 484
Aliases and LDAP 485
Sendmail Configuration 487
Sendmail Masquerading 491
Configuring Mail Servers and Mail Clients 493
Configuring Sendmail for a Simple Network Configuration 494
Configuring Sendmail for a Centralized Mail Server 494
Configuring a Workstation with Direct ISP Connection 495
The Mailer Table 495
Virtual Domains: virtusertable 496
Security 496
POP and IMAP Server: Dovecot 498
Dovecot 499
Other POP and IMAP Servers 499
Spam: SpamAssassin 500

26 Print, News, Search, and Database Servers 503
Printer Servers: CUPS 503
Printer Devices and Configuration 504
Printer Device Files 504
Spool Directories 505
Installing Printers with CUPS 505
Configuring CUPS on GNOME 505
Configuring CUPS on KDE 505
CUPS Web Browser-Based Configuration Tool 506
Configuring Remote Printers on CUPS 507
CUPS Printer Classes 507
CUPS Configuration 508
cupsd.conf 508
CUPS Directives 508
CUPS Command Line Print Clients 509
lpr 509
lpc 510
lpq and lpstat 510
lprm 510
CUPS Command Line Administrative Tools 510
lpadmin 511
lpoptions 511
enable and disable 512
accept and reject 512
lpinfo 512
News Servers 512
News Servers: INN 513
Newsreader Access 514
Overviews 514
INN Implementation 515
Database Servers: MySQL and PostgreSQL 515
Relational Database Structure 516
SQL 516

MySQL 517
PostgreSQL 520

Part VII System Administration

27 Basic System Administration 523
Superuser Control: The Root User 523
Root User Password 524
Root User Access: su 524
Controlled Administrative Access: sudo 525
System Time and Date 526
Scheduling Tasks: cron 527
crontab Entries 527
Environment Variables for cron 528
The cron.d Directory 528
The crontab Command 529
Editing in cron 529
Organizing Scheduled Tasks 529
Running cron Directory Scripts 530
cron Directory Names 531
Anacron 531
System Runlevels: telinit, initab, and shutdown 531
Runlevels 531
Runlevels in initab 533
Changing Runlevels with telinit 533
The runlevel Command 534
Shutdown 534
System Directories 536
Program Directories 537
Configuration Directories and Files 537
Configuration Files: /etc 537
System Logs: /var/log and syslogd 537
syslogd and syslog.conf 537
Entries in syslog.conf 539
Priorities 540
Actions and Users 540
An Example for /etc/syslog.conf 541
The Linux Auditing System: auditd 541
Performance Analysis Tools and Processes 542
GNOME System Monitor 543
The ps Command 543
vmstat, top, free, Xload, iostat, and sar 544
System Tap 544
Frysk 544
GNOME Power Manager 545
GKrellM 545
KDE Task Manager and Performance Monitor (KSysguard) 546
Grand Unified Bootloader (GRUB) 547

28 Managing Users **551**
GUI User Managment Tools: users-admin and KUser 551
User Configuration Files 552
The Password Files 553
/etc/passwd 553
/etc/shadow and /etc/gshadow 554
Password Tools 554
Managing User Environments 554
Profile Scripts 554
/etc/skel 555
/etc/login.defs 555
/etc/login.access 555
Controlling User Passwords 556
Adding and Removing Users with useradd, usermod, and userdel 557
useradd 558
usermod 559
userdel 559
Managing Groups 559
/etc/group and /etc/gshadow 559
User Private Groups 560
Group Directories 560
Managing Groups Using groupadd, groupmod, and groupdel 561
Controlling Access to Directories and Files: chmod 561
Permissions 561
chmod 563
Ownership 563
Changing a File's Owner or Group: chown and chgrp 565
Setting Permissions: Permission Symbols 566
Absolute Permissions: Binary Masks 566
Directory Permissions 568
Ownership Permissions 569
Sticky Bit Permissions 569
Permission Defaults: umask 570
Disk Quotas 571
Quota Tools 571
edquota 571
quotacheck, quotaon, and quotaoff 572
repquota and quota 572
Lightweight Directory Access Protocol 573
LDAP Clients and Servers 573
LDAP Configuration Files 574
Configuring the LDAP server: /etc/slapd.conf 574
LDAP Directory Database: ldif 575
LDAP Tools 579
LDAP and PAM 580
LDAP and the Name Service Switch Service 580
Pluggable Authentication Modules 580
PAM Configuration Files 581
PAM Modules 581

29 File Systems 583
File Systems 584
File System Hierarchy Standard (FHS) 584
Root Directory: / 584
System Directories 585
The /usr Directory 587
The /media Directory 587
The /mnt Directory 587
The /home Directory 588
The /var Directory 588
The /proc File System 589
The sysfs File System: /sys 589
Device Files: /dev, udev, and HAL 590
Mounting File Systems 593
File System Information 593
Journaling 594
ext3 Journaling 595
ReiserFS 595
Mounting File Systems Automatically: /etc/fstab 596
HAL and fstab 596
fstab Fields 596
Auto Mounts 598
mount Options 598
Boot and Disk Check 598
fstab Sample 599
Partition Labels: e2label 600
Windows Partitions 600
Linux Kernel Interfaces 601
noauto 601
Mounting File Systems Manually: mount and umount 601
The mount Command 602
The umount Command 603
Mounting Floppy Disks 604
Mounting CD-ROMs 604
Mounting Hard Drive Partitions: Linux and Windows 605
Creating File Systems: mkfs, mke2fs, mkswap, parted, and fdisk 606
fdisk 606
parted 608
mkfs 609
mkswap 610
CD-ROM and DVD-ROM Recording 610
mkisofs 611
cddrecord 612
DVD+RW Tools 613
Mono and .NET Support 613

30 RAID and LVM 615
Logical Volume Manager (LVM) 616
LVM Structure 616
Creating LVMs During Installation 617

Distribution Configuration Tools 617
LVM Tools: Using the LVM Commands 617
Using LVM to Replace Drives 622
LVM Example for Partitions on Different Hard Drives 623
LVM Snapshots 625
Configuring RAID Devices 625
Motherboard RAID Support: dmraid 626
Linux Software RAID Levels 627
RAID Devices and Partitions: md and fd 629
Booting from a RAID Device 629
RAID Administration: mdadm 629
Creating and Installing RAID Devices 630
Corresponding Hard Disk Partitions 635
RAID Example 636

31 Devices and Modules 639
The sysfs File System: /sys 639
The proc File System: /proc 641
udev: Device Files 641
udev Configuration 642
Device Names and udev Rules: /etc/udev/rules.d 643
Symbolic Links 645
Program Fields, IMPORT{program} keys, and /lib/udev 648
Creating udev Rules 648
SYMLINK Rules 649
Persistent Names: udevinfo 650
Hardware Abstraction Layer: HAL 652
The HAL Daemon and hal-device-manager (hal-gnome) 653
HAL Configuration: /etc/hal/fdi, and /usr/share/hal/fdi 653
Device Information Files: fdi 654
Properties 654
Device Information File Directives 656
Manual Devices 657
Device Types 658
MAKEDEV 658
mknod 659
Installing and Managing Terminals and Modems 660
Serial Ports 660
mingetty, mgetty, and agetty 661
termcap and inittab Files 661
tset 661
Input Devices 662
Installing Sound, Network, and Other Cards 662
Sound Devices 662
Video and TV Devices 663
PCMCIA Devices 664
Modules 664
Kernel Module Tools 664
Module Files and Directories: /lib/modules 665
Managing Modules with modprobe 666

The depmod Command 666
The modprobe Command 666
The insmod Command 667
The rmmod Command 667
modprobe configuration 667
Installing New Modules from Vendors: Driver Packages 669
Installing New Modules from the Kernel 670

32 Kernel Administration 671
Kernel Versions 671
References 672
Kernel Tuning: Kernel Runtime Parameters 673
Installing a New Kernel Version 673
CPU Kernel Packages 674
Installing Kernel Packages: /boot 674
Precautionary Steps for Modifying a Kernel of the Same Version 675
Boot Loader 675
Compiling the Kernel from Source Code 676
Installing Kernel Sources: Kernel Archives and Patches 677
Configuring the Kernel 677
Kernel Configuration Tools 677
Important Kernel Configuration Features 679
Compiling and Installing the Kernel 681
Installing the Kernel Image Manually 682
Kernel Boot Disks 683
Boot Loader Configurations: GRUB 684
Module RAM Disks 684
Virtualization 685
Virtual Machine Manager: virt-manager (Red Hat) 686
Kernel-Based Virtualization Machine (KVM): Hardware Virtualization 687
Xen Virtualization Kernel 688

33 Backup Management 693
Individual Backups: archive and rsync 693
BackupPC 694
Amanda 695
Amanda Commands 695
Amanda Configuration 695
Enabling Amanda on the Network 697
Using Amanda 697
Backups with dump and restore 698
The dump Levels 698
Recording Backups 700
Operations with dump 700
Recovering Backups 701

Part VIII Network Administration Services

34 Administering TCP/IP Networks 707
TCP/IP Protocol Suite 707

Configuring Networks on GNOME and KDE 710
Zero Configuration Networking (zeroconf): Avahi and Link Local Addressing 710
IPv4 and IPv6 711
TCP/IP Network Addresses 712
IPv4 Network Addresses 712
Class-Based IP Addressing 712
Netmask 713
Classless Interdomain Routing (CIDR) 714
Obtaining an IP Address 717
Broadcast Addresses 719
Gateway Addresses 719
Name Server Addresses 719
IPv6 Addressing 720
IPv6 Address Format 720
IPv6 Interface Identifiers 721
IPv6 Address Types 721
IPv6 and IPv4 Coexistence Methods 723
TCP/IP Configuration Files 723
Identifying Hostnames: /etc/hosts 723
/etc/resolv.conf 725
/etc/services 725
/etc/protocols 725
Domain Name Service (DNS) 725
host.conf 726
/etc/nsswitch.conf: Name Service Switch 727
Network Interfaces and Routes: ifconfig and route 729
ifconfig 729
Routing 731
Wireless Networking 733
Network Manager: GNOME 733
Manual Wireless Configurations 735
Command Line PPP Access: wvdial 737
Monitoring Your Network: ping, netstat, tcpdump, EtherApe, Ettercap, and Wireshark 739
ping 739
Ettercap 739
Wireshark 739
tcpdump 741
netstat 742
IP Aliasing 742
InfiniBand Support 743

35 Network Autoconfiguration with IPv6, DHCPv6, and DHCP 745
IPv6 Stateless Autoconfiguration 745
Generating the Local Address 746
Generating the Full Address: Router Advertisements 746
Router Renumbering 746
IPv6 Stateful Autoconfiguration: DHCPv6 748
Linux as an IPv6 Router: radvd 749

DHCP for IPv4 750
Configuring DHCP IPv4 Client Hosts 750
Configuring the DHCP IPv4 Server 751
Dynamic IPv4 Addresses for DHCP 754
DHCP Dynamic DNS Updates 755
DHCP Subnetworks 757
DHCP Fixed Addresses 759

36 NFS and NIS 761
Network File Systems: NFS and /etc/exports 761
NFSv4 761
NFS Daemons 762
Starting and Stopping NFS 762
NFS Configuration: /etc/exports 762
NFS File and Directory Security with NFS4 Access Lists 766
Controlling Accessing to NFS Servers 766
Mounting NFS File Systems: NFS Clients 768
Network Information Service: NIS 770
NIS Servers 771
Netgroups 774
NIS Clients 774

37 Distributed Network File Systems 777
Parallel Virtual File System (PVFS) 777
Coda 778
Red Hat Global File System (GFS and GFS 2) 779
GFS 2 Packages (Fedora Core 6 and On) 780
GFS 2 Service Scripts 780
Implementing a GFS 2 File System 781
GFS Tools 781
GFS File System Operations 783
GFS 1 784

A Where to Obtain Linux Distributions 785

Index 787

www.allitebooks.com

Acknowledgments

I would like to thank all those at McGraw-Hill who made this book a reality, particularly Jane Brownlow, sponsoring editor, for her continued encouragement and analysis as well as management of such a complex project; Dean Henrichsmeyer, the technical editor, whose analysis and suggestions proved very insightful and helpful; Jennifer Housh, acquisitions coordinator, who provided needed resources and helpful advice; Sally Engelfried, copy editor, for her excellent job editing as well as insightful comments; project manager, Sam RC who, along with editorial manager, Patty Mon, incorporated the large number of features found in this book as well as coordinated the intricate task of generating the final version. Thanks also to Scott Rogers, who initiated the project.

Special thanks to Linus Torvalds, the creator of Linux, and to those who continue to develop Linux as an open, professional, and effective operating system accessible to anyone. Thanks also to the academic community whose special dedication has developed Unix as a flexible and versatile operating system. I would also like to thank professors and students at the University of California, Berkeley, for the experience and support in developing new and different ways of understanding operating system technologies.

I would also like to thank my parents, George and Cecelia, and my brothers, George, Robert, and Mark, for their support and encouragement of such a difficult project. Also Valerie and Marylou and my nieces and nephews, Aleina, Larisa, Justin, Christopher, and Dylan, for their support and deadline reminders.

This page intentionally left blank

Introduction

The Linux operating system has become one of the major operating systems in use today, bringing to the PC all the power and flexibility of a Unix workstation as well as a complete set of Internet applications and a fully functional desktop interface. This book is designed not only to be a complete reference on Linux, but also to provide clear and detailed explanations of Linux features. No prior knowledge of Unix is assumed; Linux is an operating system anyone can use.

With the large number of Linux distributions available, it is easy to lose sight of the fact that most of their operations are the same. They all use the same desktops, shell, file systems, servers, administration support, and network configurations. Many distributions provide their own GUI tools, but these are just front ends to the same underlying Linux commands. This book is distribution independent, providing a concise and detailed explanation of those tasks common to all Linux systems. As much as 95 percent of a Linux system involves operations that are the same for all distributions. You can use this book no matter what particular Linux distribution you are using.

Linux distributions include features that have become standard, like the desktops; Unix compatibility; network servers; and numerous software applications such as office, multimedia, and Internet applications. GNOME and the K Desktop Environment (KDE) have become standard desktop Graphical User Interfaces (GUI) for Linux, noted for their power, flexibility, and ease of use. Both have become integrated components of Linux, with applications and tools for every kind of task and operation.

Linux is also a fully functional Unix operating system. It has all the standard features of a powerful Unix system, including a complete set of Unix shells such as BASH, TCSH, and the Z shell. Those familiar with the Unix interface can use any of these shells, with the same Unix commands, filters, and configuration features.

A wide array of applications operate on Linux. Numerous desktop applications are continually released on the distribution repositories. The GNU General Public License (GPL) software provides professional-level applications such as programming development tools, editors, and word processors, as well as numerous specialized applications such as those for graphics and sound.

How to Use This Book

This book identifies seven major Linux topics: shell environments, desktops, applications, security, servers, system administration, and network administration. It is really several books in one—a desktop book, a shell-user book, a security book, a server book, and an administration book—how you choose to use it depends upon how you want to use your

Linux system. Almost all Linux operations can be carried out using either the GNOME or KDE interface. You can focus on the GNOME and KDE chapters and their corresponding tools and applications in the different chapters throughout the book. On the other hand, if you want to delve deeper into the Unix aspects of Linux, you can check out the shell chapters and the corresponding shell-based applications in other chapters. If you only want to use Linux for its applications and Internet clients, then concentrate on the applications section. If you want to use Linux as a multiuser system servicing many users or integrate it into a local network, you can use the detailed system, file, and network administration information provided in the administration chapters. None of these tasks are in any way exclusive. If you are working in a business environment, you will probably make use of all three aspects. Single users may concentrate more on the desktops and applications, whereas administrators may make more use of the security and networking features.

Part Topics

The first part of this book provides a general overview and covers some startup topics that users may find helpful. It provides an introduction to Linux listings of resources, software sites, documentation sites, newsgroups and Linux news and development sites. Distributions are covered briefly. The next chapter covers startup topics such as general install issues, GNOME and KDE basics, as well as Windows access.

Part II of this book deals with Linux shell environments, covering the BASH and TCSH shells, shell scripts, shell configuration, and the Linux file system. All these chapters operate from a command line interface, letting you access and manage files and shells directly.

Part III of this book covers desktops and their GUI support tools like the X Window System and display managers. Here you are introduced to the KDE and GNOME desktops. Different features such as applets, the Panel, and configuration tools are described in detail.

Part IV of this book discusses in detail the many office, multimedia, and Internet applications you can use on your Linux system, beginning with office suites like OpenOffice .org and KOffice. The different database management systems available are also discussed, along with the website locations where you can download them. Linux automatically installs mail, news, FTP, and web browser applications, as well as FTP and web servers. Both KDE and GNOME come with a full set of mail, news, FTP clients and web browsers.

Part V demonstrates how to implement security precautions using encryption, authentication, and firewalls. Coverage of the GNU Privacy Guard (GPG) shows you how to implement public- and private key-based encryption. With Luks (Linux Unified Key Setup) you can easily encrypt file systems. SE Linux provides comprehensive and refined control of all your network and system resources. IPsec tools let you use the IPSEC protocol to encrypt and authentication network transmissions. Network security topics cover firewalls and encryption using Netfilter (IPtables) to protect your system, the Secure Shell (SSH) to provide secure remote transmissions, and Kerberos to provide secure authentication.

Part VI discusses Internet servers you can run on Linux, including FTP, web, and mail servers. The Apache web server chapter covers standard configuration directives like those for automatic indexing as well as the newer virtual host directives. Sendmail, Postfix, IMAP, and POP mail servers are also covered, and the INN news server, the CUPS print server, the MySQL database server, and the Squid proxy server are examined.

Part VII discusses system administration topics including user, software, file system, system, device, and kernel administration. There are detailed descriptions of the configuration files used in administration tasks and how to make entries in them. First, basic system

administration tasks are covered, such as selecting runlevels, monitoring your system, and scheduling shutdowns. Then, aspects of setting up and controlling users and groups are discussed. Different methods of virtualization are covered, such as full (KVM) and para-virtualizaton (Xen). Different file system tasks are covered, such as mounting file systems, managing file systems with HAL and udev, and configuring RAID devices and LVM volumes. Devices are automatically detected with udev and the Hardware Abstraction Layer (HAL).

Part VIII covers network administration topics such as configuring network interfaces and IP addressing.You also learn how to implement your own IPv4 Dynamic Host Configuration Protocol (DHCP) server to dynamically assign hosts IP addresses and how IPv6 automatic addressing and renumbering operates. The various network file system (NFS) interfaces and services such as GFS version 2, NFS for Unix, and NIS networks are presented.

This page intentionally left blank

PART

Introduction

CHAPTER 1
Introduction to Linux
CHAPTER 2
Getting Started

This page intentionally left blank

CHAPTER 1

Introduction to Linux

Linux is a fast and stable open source operating system for personal computers (PCs) and workstations that features professional-level Internet services, extensive development tools, fully functional graphical user interfaces (GUIs), and a massive number of applications ranging from office suites to multimedia applications. Linux was developed in the early 1990s by Linus Torvalds, along with other programmers around the world. As an operating system, Linux performs many of the same functions as Unix, Macintosh, Windows, and Windows NT. However, Linux is distinguished by its power and flexibility, along with being freely available. Most PC operating systems, such as Windows, began their development within the confines of small, restricted PCs, which have only recently become more versatile machines. Such operating systems are constantly being upgraded to keep up with the ever-changing capabilities of PC hardware. Linux, on the other hand, was developed in a different context. Linux is a PC version of the Unix operating system that has been used for decades on mainframes and minicomputers and is currently the system of choice for network servers and workstations. Linux brings the speed, efficiency, scalability, and flexibility of Unix to your PC, taking advantage of all the capabilities that PCs can now provide.

Technically, Linux consists of the operating system program, referred to as the *kernel,* which is the part originally developed by Linus Torvalds. But it has always been distributed with a massive number of software applications, ranging from network servers and security programs to office applications and development tools. Linux has evolved as part of the open source software movement, in which independent programmers joined together to provide free, high-quality software to any user. Linux has become the premier platform for open source software, much of it developed by the Free Software Foundation's GNU project. Many of these applications are bundled as part of standard Linux distributions. Currently, thousands of open source applications are available for Linux from sites like SourceForge, Inc.'s **sourceforge.net**, K Desktop Environment's (KDE's) **kde-apps.org**, and GNU Network Object Model Environment's (GNOME's) **gnomefiles.org**. Most of these applications are also incorporated into the distribution repository, using packages that are distribution compliant.

Along with Linux's operating system capabilities come powerful networking features, including support for Internet, intranets, and Windows networking. As a norm, Linux distributions include fast, efficient, and stable Internet servers, such as the web, File Transfer Protocol (FTP), and DNS servers, along with proxy, news, and mail servers. In other words, Linux has everything you need to set up, support, and maintain a fully functional network.

With both GNOME and KDE, Linux also provides GUIs with that same level of flexibility and power. Unlike Windows and the Mac, Linux enables you to choose the interface you want and then customize it further, adding panels, applets, virtual desktops, and menus, all with full drag-and-drop capabilities and Internet-aware tools.

Linux does all this at the right price. Linux is free, including the network servers and GUI desktops. Unlike the official Unix operating system, Linux is distributed freely under a GNU general public license as specified by the Free Software Foundation, making it available to anyone who wants to use it. GNU (the acronym stands for "GNUs Not Unix") is a project initiated and managed by the Free Software Foundation to provide free software to users, programmers, and developers. Linux is copyrighted, not public domain. However, a GNU public license has much the same effect as the software's being in the public domain. The GNU GPL is designed to ensure Linux remains free and, at the same time, standardized. Linux is technically the operating system kernel—the core operations—and only one official Linux kernel exists. People sometimes have the mistaken impression that Linux is somehow less than a professional operating system because it is free. Linux is, in fact, a PC, workstation, and server version of Unix. Many consider it far more stable and much more powerful than Windows. This power and stability have made Linux an operating system of choice as a network server.

To appreciate Linux completely, you need to understand the special context in which the Unix operating system was developed. Unix, unlike most other operating systems, was developed in a research and academic environment. In universities, research laboratories, data centers, and enterprises, Unix is the system most often used. Its development has paralleled the entire computer and communications revolution over the past several decades. Computer professionals often developed new computer technologies on Unix, such as those developed for the Internet. Although a sophisticated system, Unix was designed from the beginning to be flexible. The Unix system itself can be easily modified to create different versions. In fact, many different vendors maintain different official versions of Unix. IBM, Sun, and Hewlett-Packard all sell and maintain their own versions of Unix. The unique demands of research programs often require that Unix be tailored to their own special needs. This inherent flexibility in the Unix design in no way detracts from its quality. In fact, this flexibility attests to the ruggedness of Unix, allowing it to adapt to practically any environment. This is the context in which Linux was developed. Linux is, in this sense, one other version of Unix—a version for the PC. The development of Linux by computer professionals working in a researchlike environment reflects the way Unix versions have usually been developed. Linux is publicly licensed and free—and reflects the deep roots Unix has in academic institutions, with their sense of public service and support. Linux is a top-rate operating system accessible to everyone, free of charge.

Linux Distributions

Although there is only one standard version of Linux, there are actually several different distributions. Different companies and groups have packaged Linux and Linux software in slightly different ways. Each company or group then releases the Linux package, usually on a CD-ROM. Later releases may include updated versions of programs or new software. Some of the more popular distributions are Red Hat, Ubuntu, Mepis, SUSE, Fedora, and Debian. The Linux kernel is centrally distributed through **kernel.org**. All distributions use this same kernel, although it may be configured differently.

www.allitebooks.com

Linux has spawned a great variety of distributions. Many aim to provide a comprehensive solution providing support for any and all task. These include distributions like SUSE, Red Hat, and Ubuntu. Some are variations on other distributions, like Centos, which is based on Red Hat Enterprise Linux, and Ubuntu, which derives from Debian Linux. Others have been developed for more specialized tasks or to support certain features. Distributions like Debian provide cutting edge developments. Some distributions provide more commercial versions, usually bundled with commercial applications such as databases or secure servers. Certain companies like Red Hat and Novell provide a commercial distribution that corresponds to a supported free distribution. The free distribution is used to develop new features, like the Fedora Project for Red Hat. Other distributions like Knoppix and Ubuntu specialize in Live-CDs, the entire Linux operating system on single CD.

Currently, **distrowatch.com** lists numerous Linux distributions. Check this site for details about current distributions. Table 1-1 lists the websites for several of the more popular Linux distributions. The FTP sites for these distributions use the prefix **ftp** instead of **www**, as in **ftp.redhat.com**. Also listed in Table 1-1 is the Linux kernel site where the newest releases of the official Linux kernel are provided. These sites have corresponding FTP sites where you can download updates and new releases.

NOTE *Distributions will use their own software install and update programs. Check your distribution documentation for details.*

URL	Site Description
redhat.com	Red Hat Linux
fedoraproject.org	Fedora Linux
centos.org	Centos Linux
opensuse.com	openSUSE Linux
debian.org	Debian Linux
ubuntu.com	Ubuntu Linux
mepis.org	Mepis Linux
gentoo.org	Gentoo Linux
turbolinux.com	Turbo Linux
knoppix.org	Knoppix Linux
linuxiso.com	CD-ROM ISO images of Linux distributions
distrowatch.com	Detailed information about Linux distributions
kernel.org	Linux kernel

TABLE 1-1 Linux Distribution and Kernel Sites

Operating Systems and Linux

An *operating system* is a program that manages computer hardware and software for the user. Operating systems were originally designed to perform repetitive hardware tasks, which centered around managing files, running programs, and receiving commands from the user. You interact with an operating system through a *user interface*, which allows the operating system to receive and interpret instructions sent by the user. You need only send an instruction to the operating system to perform a task, such as reading a file or printing a document. An operating system's user interface can be as simple as entering commands on a line or as complex as selecting menus and icons on a desktop.

An operating system also manages software applications. To perform different tasks, such as editing documents or performing calculations, you need specific software applications. An *editor* is an example of a software application that enables you to edit a document, making changes and adding new text. The editor itself is a program consisting of instructions to be executed by the computer. For the program to be used, it must first be loaded into computer memory, and then its instructions are executed. The operating system controls the loading and execution of all programs, including any software applications. When you want to use an editor, simply instruct the operating system to load the editor application and execute it.

File management, program management, and user interaction are traditional features common to all operating systems. Linux, like all versions of Unix, adds two more features. Linux is a multiuser and multitasking system. As it is a *multitasking* system, you can ask the system to perform several tasks at the same time. While one task is being done, you can work on another. For example, you can edit a file while another file is being printed. You do not have to wait for the other file to finish printing before you edit. As it is a *multiuser* system, several users can log in to the system at the same time, each interacting with the system through his or her own terminal.

As a version of Unix, Linux shares that system's flexibility, a flexibility stemming from Unix's research origins. Developed by Ken Thompson at AT&T Bell Laboratories in the late 1960s and early 1970s, the Unix system incorporated many new developments in operating system design. Originally, Unix was designed as an operating system for researchers. One major goal was to create a system that could support the researchers' changing demands. To do this, Thompson had to design a system that could deal with many different kinds of tasks. Flexibility became more important than hardware efficiency. Like Unix, Linux has the advantage of being able to deal with the variety of tasks any user may face. The user is not confined to limited and rigid interactions with the operating system. Instead, the operating system is thought of as making a set of highly effective tools available to the user. This user-oriented philosophy means you can configure and program the system to meet your specific needs. With Linux, the operating system becomes an operating environment.

History of Unix and Linux

As a version of Unix, the history of Linux naturally begins with Unix. The story begins in the late 1960s, when a concerted effort to develop new operating system techniques occurred. In 1968, a consortium of researchers from General Electric, AT&T Bell Laboratories, and the Massachusetts Institute of Technology carried out a special operating system research project called MULTICS (the Multiplexed Information and Computing Service). MULTICS incorporated many new concepts in multitasking, file management, and user interaction.

Unix

In 1969, Ken Thompson, Dennis Ritchie, and the researchers at AT&T Bell Laboratories developed the Unix operating system, incorporating many of the features of the MULTICS research project. They tailored the system for the needs of a research environment, designing it to run on minicomputers. From its inception, Unix was an affordable and efficient multiuser and multitasking operating system.

The Unix system became popular at Bell Labs as more and more researchers started using the system. In 1973, Dennis Ritchie collaborated with Ken Thompson to rewrite the programming code for the Unix system in the C programming language. Unix gradually grew from one person's tailored design to a standard software product distributed by many different vendors, such as Novell and IBM. Initially, Unix was treated as a research product. The first versions of Unix were distributed free to the computer science departments of many noted universities. Throughout the 1970s, Bell Labs began issuing official versions of Unix and licensing the systems to different users. One of these users was the computer science department of the University of California, Berkeley. Berkeley added many new features to the system that later became standard. In 1975 Berkeley released its own version of Unix, known by its distribution arm, Berkeley Software Distribution (BSD). This BSD version of Unix became a major contender to the AT&T Bell Labs version. AT&T developed several research versions of Unix, and in 1983 it released the first commercial version, called System 3. This was later followed by System V, which became a supported commercial software product.

At the same time, the BSD version of Unix was developing through several releases. In the late 1970s, BSD Unix became the basis of a research project by the Department of Defense's Advanced Research Projects Agency (DARPA). As a result, in 1983, Berkeley released a powerful version of Unix called BSD release 4.2. This release included sophisticated file management as well as networking features based on Internet network protocols—the same protocols now used for the Internet. BSD release 4.2 was widely distributed and adopted by many vendors, such as Sun Microsystems.

In the mid-1980s, two competing standards emerged, one based on the AT&T version of Unix and the other based on the BSD version. AT&T's Unix System Laboratories developed System V release 4. Several other companies, such as IBM and Hewlett-Packard, established the Open Software Foundation (OSF) to create their own standard version of Unix. Two commercial standard versions of Unix existed then—the OSF version and System V release 4.

Linux

Originally designed specifically for Intel-based PCs, Linux started out at the University of Helsinki as a personal project of a computer science student named Linus Torvalds. At that time, students were making use of a program called *Minix*, which highlighted different Unix features. Minix was created by Professor Andrew Tanenbaum and widely distributed over the Internet to students around the world. Linus's intention was to create an effective PC version of Unix for Minix users. It was named Linux, and in 1991, Linus released version 0.11. Linux was widely distributed over the Internet, and in the following years, other programmers refined and added to it, incorporating most of the applications and features now found in standard Unix systems. All the major window managers have been ported to Linux. Linux has all the networking tools, such as FTP support, web browsers, and the whole range of network services such as email, the domain name service, and dynamic host

configuration, along with FTP, web, and print servers. It also has a full set of program development utilities, such as C++ compilers and debuggers. Given all its features, the Linux operating system remains small, stable, and fast. In its simplest format, Linux can run effectively on only 2MB of memory.

Although Linux has developed in the free and open environment of the Internet, it adheres to official Unix standards. Because of the proliferation of Unix versions in the previous decades, the Institute of Electrical and Electronics Engineers (IEEE) developed an independent Unix standard for the American National Standards Institute (ANSI). This new ANSI-standard Unix is called the Portable Operating System Interface for Computer Environments (POSIX). The standard defines how a Unix-like system needs to operate, specifying details such as system calls and interfaces. POSIX defines a universal standard to which all Unix versions must adhere. Most popular versions of Unix are now POSIX-compliant. Linux was developed from the beginning according to the POSIX standard. Linux also adheres to the Linux file system hierarchy standard (FHS), which specifies the location of files and directories in the Linux file structure. See **pathname.com/fhs** for more details.

Linux development is now overseen by The Linux Foundation (**linux-foundation.org**), which is a merger of The Free Standards Group and Open Source Development Labs (OSDL). This is the group that Linus Torvalds works with to develop new Linux versions. Actual Linux kernels are released at **kernel.org**.

Linux Overview

Like Unix, Linux can be generally divided into three major components: the kernel, the environment, and the file structure. The *kernel* is the core program that runs programs and manages hardware devices, such as disks and printers. The *environment* provides an interface for the user. It receives commands from the user and sends those commands to the kernel for execution. The *file structure* organizes the way files are stored on a storage device, such as a disk. Files are organized into directories. Each directory may contain any number of subdirectories, each holding files. Together, the kernel, the environment, and the file structure form the basic operating system structure. With these three, you can run programs, manage files, and interact with the system.

An environment provides an interface between the kernel and the user. It can be described as an interpreter. Such an interface interprets commands entered by the user and sends them to the kernel. Linux provides several kinds of environments: desktops, window managers, and command line shells. Each user on a Linux system has his or her own user interface. Users can tailor their environments to their own special needs, whether they be shells, window managers, or desktops. In this sense, for the user, the operating system functions more as an operating environment, which the user can control.

In Linux, files are organized into directories, much as they are in Windows. The entire Linux file system is one large interconnected set of directories, each containing files. Some directories are standard directories reserved for system use. You can create your own directories for your own files, as well as easily move files from one directory to another. You can even move entire directories and share directories and files with other users on your system. With Linux, you can also set permissions on directories and files, allowing others to access them or restricting access to yourself alone. The directories of each user are, in fact, ultimately connected to the directories of other users. Directories are organized into a hierarchical tree structure, beginning with an initial root directory. All other directories are ultimately derived from this first root directory.

With KDE and GNOME, Linux now has a completely integrated GUI. You can perform all your Linux operations entirely from either interface. KDE and GNOME are fully operational desktops supporting drag-and-drop operations, enabling you to drag icons to your desktop and to set up your own menus on an Applications panel. Both rely on an underlying X Window System, which means as long as they are both installed on your system, applications from one can run on the other desktop. The GNOME and KDE sites are particularly helpful for documentation, news, and software you can download for those desktops. Both desktops can run any X Window System program, as well as any cursor-based program such as Emacs and Vi, which were designed to work in a shell environment. At the same time, a great many applications are written just for those desktops and included with your distributions. KDE and GNOME have complete sets of Internet tools, along with editors and graphics, multimedia, and system applications. Check their websites at **gnome.org** and **kde.org** for latest developments. As new versions are released, they include new software.

Open Source Software

Linux was developed as a cooperative open source effort over the Internet, so no company or institution controls Linux. Software developed for Linux reflects this background. Development often takes place when Linux users decide to work on a project together. The software is posted at an Internet site, and any Linux user can then access the site and download the software. Linux software development has always operated in an Internet environment and is global in scope, enlisting programmers from around the world. The only thing you need to start a Linux-based software project is a website.

Most Linux software is developed as open source software. This means that the source code for an application is freely distributed along with the application. Programmers over the Internet can make their own contributions to a software package's development, modifying and correcting the source code. Linux is an open source operating system as well. Its source code is included in all its distributions and is freely available on the Internet. Many major software development efforts are also open source projects, as are the KDE and GNOME desktops, along with most of their applications. The Netscape Communicator web browser package has also become open source, with its source code freely available. The OpenOffice office suite supported by Sun is an open source project based on the StarOffice office suite (StarOffice is essentially Sun's commercial version of OpenOffice). Many of the open source applications that run on Linux have located their websites at SourceForge (**sourceforge.net**), which is a hosting site designed specifically to support open source projects. You can find more information about the open source movement at **opensource.org**.

Open source software is protected by public licenses. These prevent commercial companies from taking control of open source software by adding a few modifications of their own, copyrighting those changes, and selling the software as their own product. The most popular public license is the GNU GPL provided by the Free Software Foundation. This is the license that Linux is distributed under. The GNU GPL retains the copyright, freely licensing the software with the requirement that the software and any modifications made to it always be freely available. Other public licenses have also been created to support the demands of different kinds of open source projects. The GNU lesser general public license (LGPL) lets commercial applications use GNU licensed software libraries. The qt public license (QPL) lets open source developers use the Qt libraries essential to the KDE desktop. You can find a complete listing at **opensource.org**.

Linux is currently copyrighted under a GNU public license provided by the Free Software Foundation, and it is often referred to as GNU software (see **gnu.org**). GNU software is distributed free, provided it is freely distributed to others. GNU software has proved both reliable and effective. Many of the popular Linux utilities, such as C compilers, shells, and editors, are GNU software applications. Installed with your Linux distribution are the GNU C++ and Lisp compilers, Vi and Emacs editors, BASH and TCSH shells, as well as TeX and Ghostscript document formatters. In addition, there are many open source software projects that are licensed under the GNU GPL.

Under the terms of the GNU GPL, the original author retains the copyright, although anyone can modify the software and redistribute it, provided the source code is included, made public, and provided free. Also, no restriction exists on selling the software or giving it away free. One distributor could charge for the software, while another one could provide it free of charge. Major software companies are also providing Linux versions of their most popular applications. Oracle provides a Linux version of its Oracle database. (At present, no plans seem in the works for Microsoft applications.)

Linux Software

All Linux software is currently available from online repositories. You can download applications for desktops, Internet servers, office suites, and programming packages, among others. Software packages may be distributed through online repositories. Downloads and updates are handled automatically by your desktop software manager and updater.

In addition, you can download from third-party sources software that is in the form of compressed archives or software packages like RPM and DEB. RPM packages are those archived using the Red Hat Package Manager, which is used on several distributions. Compressed archives have an extension such as **.tar.gz** or **.tar.Z**, whereas RPM packages have an **.rpm** extension and DEB uses a **.deb** extension. Any RPM package that you download directly, from whatever site, can be installed easily with the click of a button using a distribution software manager on a desktop. You can also download the source version and compile it directly on your system. This has become a simple process, almost as simple as installing the compiled RPM versions.

Linux distributions also have a large number of mirror sites from which you can download their software packages for current releases. If you have trouble connecting to a main FTP site, try one of its mirrors.

Software Repositories

For many distributions, you can update to the latest software from the online repositories using a software updater. Linux distributions provide a comprehensive selection of software ranging from office and multimedia applications to Internet servers and administration services. Many popular applications are not included, though they may be provided on associated software sites. During installation, your software installer is configured to access your distribution repository.

Because of licensing restrictions, multimedia support for popular formats like MP3, DVD, and DivX is not included with distributions. A distribution-associated site, however, may provide support for these functions, and from there you can download support for MP3, DVD, and DivX software. You can download a free licensed MP3 gstreamer plug-in

from **fluendo.com**, for example. Many distributions do not provide support for the official Nvidia- or ATI-released Linux graphics drivers, but support for these can be found at associated distribution sites. Linux distributions do include the generic X.org Nvidia and ATI drivers, which will enable your graphics cards to work.

Third-Party Linux Software Repositories

Though almost all applications should be included in the distribution software repository, you could download and install software from third-party repositories. Always check first to see if the software you want is already in the distribution repository. If it is not available, then download from a third-party repository.

Several third-party repositories make it easy to locate an application and find information about it. Of particular note are **sourceforge.net**, **rpmfind.net**, **gnomefiles.org**, and **kde-apps.org**. The following tables list different sites for Linux software. Some third-party repositories and archives for Linux software are listed in Table 1-2, along with several specialized sites, such as those for commercial and game software. When downloading software packages, always check to see if versions are packaged for your particular distribution.

Linux Office and Database Software

Many professional-level databases and office suites are now available for Linux. These include Oracle and IBM databases, as well as the OpenOffice and KOffice suites. Table 1-3 lists sites for office suites and databases. Most of the office suites, as well as MySQL and PostgreSQL, are already included on the distribution repositories and may be part of your install disk. Many of the other sites provide free personal versions of their software for Linux, and others are entirely free. You can download from them directly and install the software on your Linux system.

URL	Site Description
sourceforge.net	Lists open source software development sites for Linux applications and software repositories
jpackage.org	Repository for Java applications and tools
gnomefiles.org	GNOME applications
kde-apps.org	KDE software repository
freshmeat.net	New Linux software
rpmfind.net	RPM package repository
gnu.org	GNU archive
happypenguin.org	Linux Game Tome
linuxgames.com	Linux games
fluendo.com	Gstreamer (GNOME) multimedia licensed codecs and plug-ins (MP3, MPEG2, and so on)

TABLE 1-2 Third-Party Linux Software Archives, Repositories, and Links

URL	Software
Database Software	
oracle.com	Oracle
sybase.com	Sybase
software.ibm.com/data/db2/linux	IBM DB2
mysql.com	MySQL
ispras.ru/~kml/gss	GNU SQL
postgresql.org	PostgreSQL
Office Software	
openoffice.org	OpenOffice
koffice.kde.org	KOffice
sun.com/software/star/staroffice	StarOffice
gnomefiles.org	GNOME Office and productivity applications

TABLE 1-3 Database and Office Software

Internet Servers

One of the most important features of Linux, as of all Unix systems, is its set of Internet clients and servers. The Internet was designed and developed on Unix systems, and Internet clients and servers, such as those for FTP and the Web, were first implemented on BSD versions of Unix. DARPANET, the precursor to the Internet, was set up to link Unix systems at different universities across the nation. Linux contains a full set of Internet clients and servers, including mail, news, FTP, and web, as well as proxy clients and servers. Sites for network server and security software available for Linux are listed in Table 1-4. All of

URL	Software Description
apache.org	Apache web server
vsftpd.beasts.org	Very secure FTP server
proftpd.org	ProFTPD FTP server
isc.org	Internet Software Consortium: BIND, INN, and DHCPD
sendmail.org	Sendmail mail server
postfix.org	Postfix mail server
squid-cache.org	Squid proxy server
samba.org	Samba SMB (Windows network) server
netfilter.org	IP Tables firewall
web.mit.edu/kerberos/www	Kerberos network authentication protocol
openssh.com	Open Secure Shell (free version of SSH)

TABLE 1-4 Network Server and Security Software

URL	Site Description
gnu.org	Linux compilers and tools (gcc)
java.sun.com	Sun Java website
perl.com	Perl website with Perl software for Linux
developer.gnome.org	Website for GNOME developers
developer.kde.org	KDE library for developers

TABLE 1-5 Linux Programming Sites

these are already included on most distribution repositories and may be part of your install disk; however, you can obtain news and documentation directly from the server's websites.

Development Resources

Linux has always provided strong support for programming languages and tools. All distributions include the GNU C and C++ (gcc) compiler with supporting tools such as make. Linux distributions usually come with full development support for the KDE and GNOME desktops, letting you create your own GNOME and KDE applications. You can also download the Linux version of the Java Software Development Kit for creating Java programs. A version of Perl for Linux is also included with most distributions. You can download current versions from their websites. Table 1-5 lists different sites of interest for Linux programming.

Online Linux Information Sources

Extensive online resources are available on almost any Linux topic. The tables in this chapter list sites where you can obtain software, display documentation, and read articles on the latest developments. Many Linux websites provide news, articles, and information about Linux. Several, such as **linuxjournal.com**, are based on popular Linux magazines. Some specialize in particular areas such as **linuxgames.com** for the latest games ported for Linux. Currently, many Linux websites provide news, information, and articles on Linux developments, as well as documentation, software links, and other resources. These are listed in Table 1-6.

Linux Documentation

Linux documentation has also been developed over the Internet. Much of the documentation currently available for Linux can be downloaded from Internet FTP sites. A special Linux project called the Linux Documentation Project (LDP), headed by Matt Welsh, has developed a complete set of Linux manuals. The documentation is available at the LDP home site, **tldp.org**. Linux documents provided by the LDP are listed in Table 1-7, along with their Internet sites. The Linux documentation for your installed software will be available at your **/usr/share/doc** directory.

An extensive number of mirrors are maintained for the LDP. You can link to any of them through a variety of sources, such as the LDP home site, **tldp.org**, and **linuxjournal.org**. The documentation includes a user's guide, an introduction, and administrative guides.

URL	Site Description
tldp.org	Linux Documentation Project
lwn.net	Linux Weekly News
linux.com	Linux.com
linuxtoday.com	Linux Today
linuxplanet.com	LinuxPlanet
linuxfocus.org	Linux Focus
linuxjournal.com	Linux Journal
linuxgazette.com	Linux Gazette
linux.org	Linux Online
slashdot.org	Linux forum
opensource.org	Open source information

TABLE 1-6 Linux Information and News Sites

These are available in text, PostScript, or web page format. You can also find briefer explanations in what are referred to as HOW-TO documents.

Distribution websites provide extensive Linux documentation and software. The **gnome.org** site holds documentation for the GNOME desktop, while **kde.org** holds documentation for the KDE desktop. The tables in this chapter list many of the available sites. You can find other sites through resource pages that hold links to other websites—for example, the Linux website on the World Wide Web at **tldp.org/links.html**.

Sites	Websites
tldp.org	LDP website
Guides	**Document Format**
Linux Installation and Getting Started Guide	DVI, PostScript, LaTeX, PDF, and HTML
Linux User's Guide	DVI, PostScript, HTML, LaTeX, and PDF
Linux System Administrator's Guide	PostScript, PDF, LaTeX, and HTML
Linux Network Administrator's Guide	DVI, PostScript, PDF, and HTML
Linux Programmer's Guide	DVI, PostScript, PDF, LaTeX, and HTML
The Linux Kernel	HTML, LaTeX, DVI, and PostScript
Linux Kernel Hacker's Guide	DVI, PostScript, and HTML
Linux HOW-TOs	HTML, PostScript, SGML, and DVI
Linux FAQs	HTML, PostScript, and DVI
Linux Man Pages	Man page

TABLE 1-7 Linux Documentation Project

www.allitebooks.com

In addition to websites, Linux Usenet newsgroups are also available. Through your Internet connection, you can access Linux newsgroups to read the comments of other Linux users and to post messages of your own. Several Linux newsgroups exist, each beginning with **comp.os.linux**. One of particular interest to the beginner is **comp.os.linux.help**, where you can post questions. Table 1-8 lists some of the Linux Usenet newsgroups you can check out, particularly for posting questions.

Newsgroup	Description
comp.os.linux.announce	Announcements of Linux developments
comp.os.linux.development.apps	For programmers developing Linux applications
comp.os.linux.development.system	For programmers working on the Linux operating system
comp.os.linux.hardware	Linux hardware specifications
comp.os.linux.admin	System administration questions
comp.os.linux.misc	Special questions and issues
comp.os.linux.setup	Installation problems
comp.os.linux.answers	Answers to command problems
comp.os.linux.help	Questions and answers for particular problems
comp.os.linux.networking	Linux network questions and issues
linux.dev.group	Numerous development newsgroups beginning with **linux.dev**, such as **linux.dev.admin** and **linux.dev.doc**

TABLE 1-8 Linux Usenet Newsgroups

This page intentionally left blank

CHAPTER 2

Getting Started

Using Linux has become an intuitive process, with easy-to-use interfaces, including graphical logins and graphical user interfaces (GUIs) like GNOME and KDE. Even the standard Linux command line interface has become more user friendly with editable commands, history lists, and cursor-based tools. Distribution installation tools also use simple GUIs. Installation has become a very easy procedure, taking only a few minutes. The use of online repositories by many distributions allows for small initial installs that can be later enhanced with selected additional software.

To start using Linux, you have to know how to access your Linux system and, once you are on the system, how to execute commands and run applications. Access is supported through either the default graphical login or a command line login. For the graphical login, a simple window appears with menus for selecting login options and text boxes for entering your username and password. Once you access your system, you can then interact with it using either a command line interface or a GUI. With GUIs like GNOME and KDE, you can use windows, menus, and icons to interact with your system.

Linux is noted for providing easy access to extensive help documentation. It's easy to obtain information quickly about any Linux command and utility while logged in to the system. You can access an online manual that describes each command or obtain help that provides more detailed explanations of different Linux features. A complete set of manuals provided by the Linux Documentation Project (LDP) is on your system and available for you to browse through or print. Both the GNOME and KDE desktops provide help systems that give you easy access to desktop, system, and application help files.

Install Issues

Each distribution has its own graphical install tool that lets you install Linux very easily. Installation is often a simple matter of clicking a few buttons. However, install CDs and DVDs provide only a core subset of what is available because the software available has grown so massive that most distributions provide online repositories for downloading. Installation is now more a matter of setting up an initial configuration that you can later expand using these online repositories. Many distributions also allow you to create your

own install discs, customizing the collection of software you want on your install CD/DVD. Other installation considerations include the following:

- Most distributions provide Live-CDs that allows you to do minimal installs. This helps you avoid a lengthy download of install CDs or DVDs. You can then install just the packages you want from online repositories.
- The use of online repositories means that most installed software needs to be downloaded and updated from the repositories soon after installation. The software on install CDs and DVDs quickly becomes out of date.
- Some distributions provide updated versions of a release, including updated software since the original release. These are often provided by separate distribution projects. Check the distribution sites for availability.
- Much of your hardware is now automatically detected, including your graphics card and monitor.
- Most distributions use Parted to set up your partitions. Parted is a very easy-to-use partition management tool.
- Installation can be performed from numerous sources, by using network methods like NFS, File Transfer Protocol (FTP), and Hypertext Transfer Protocol (HTTP).
- Dual-boot installation is supported with either the GRUB or Linux Loader (LILO) boot managers. Linux boot managers can be configured easily to boot Windows, Mac, and other Linux installations on the same system.
- Distributions distinguish between 32-bit and 64-bit releases. Most CPUs in newer computers support 64-bit, whereas older or weaker systems may not.
- Network configuration is normally automatic, using Dynamic Host Configuration Protocol (DHCP) or IPv6 to connect to a network router.
- During installation you may have the option to customize your partitions, letting you set up RAID and LVM file systems if you wish.
- If you are using LVM or RAID file systems, be sure you have a separate boot partition of a standard Linux file system type.
- Most distributions perform a post-install procedure that perform basic configuration tasks like setting the date and time, configuring your firewall, and creating a user account (a root [administrative] account is set up during installation).

Most distributions provide a means to access your Linux system in rescue mode. Should your system stop working, you can access your files by using your install disc to start up Linux with a command line interface and access your installed file system. This allows you to fix your problem by editing or replacing configuration files (useful for X Window System problems with **/etc/X11/xorg.conf**).

If you have problems with the GRUB boot loader you can reinstall it with the **grub-install** command. This can happen if you later install Windows on your system. Windows will overwrite your boot manager. Use **grub-install** with the device name of the hard disk to reinstall the Linux boot manager. Be sure to put in an entry for your Windows system. Keep in mind that some distribution use alternative boot loaders like LILO.

Accessing Your Linux System

To access and use your Linux system, you must carefully follow required startup and shutdown procedures. You do not simply turn off your computer. Linux does, however, implement journaling, which allows you to automatically recover your system after the computer suddenly loses power and shuts off.

If you have installed the boot loader GRUB, when you turn on or reset your computer, the boot loader first decides what operating system to load and run. GRUB will display a menu of operating systems from which to choose.

If, instead, you wait a moment or press the ENTER key, the boot loader loads the default operating system. If a Windows system is listed, you can choose to start that instead.

You can think of your Linux operating system as operating on two different levels, one running on top of the other. The first level is when you start your Linux system and where the system loads and runs. It has control of your computer and all its peripherals. You still are not able to interact with it, however. After Linux starts, it displays a login screen, waiting for a user to log in to the system and start using it. You cannot gain access to Linux unless you log in first.

You can think of logging in and using Linux as the next level. Now you can issue commands instructing Linux to perform tasks. You can use utilities and programs such as editors or compilers, or even games. Depending on a choice you made during installation, however, you may be interacting with the system using either a simple command line interface or the desktop directly. There are both command line login prompts and graphical login windows. Most distributions will use a graphical interface by default, presenting you with a graphical login window at which you enter your username and password. If you choose not to use the graphical interface, you are presented with a simple command line prompt to enter your username.

The Display Managers: GDM and KDM

With the graphical login, your GUI starts up immediately and displays a login window with boxes for a username and password. When you enter your username and password and then press ENTER, your default GUI starts up.

For most distributions, graphical logins are handled either by the GNOME Display Manager (GDM) or the KDE Display Manager (KDM). The GDM and KDM manage the login interface along with authenticating a user password and username and then starting up a selected desktop. If problems ever occur using the GUI, you can force an exit of the GUI with the CTRL-ALT-BACKSPACE keys, returning to the login screen (or the command line if you started your GUI from there). Also, from the display manager, you can shift to the command line interface with the CTRL-ALT-F1 keys and then shift back to the GUI with the CTRL-ALT-F7 keys.

When you log out from the desktop, you return to the display manager Login window. From the Options menu, you can select the desktop or window manager you want to start up. Here you can select KDE to start up the K Desktop, for example, instead of GNOME. The Language menu lists a variety of different languages that Linux supports. Choose one to change the language interface.

To shut down your Linux system, click the Shutdown button. To restart, select the Restart option from the Options menu. Alternatively, you can also shut down or restart from your desktop. From the System menu, select the Shutdown entry. GNOME will display a dialog screen with the buttons Suspend, Shutdown, and Reboot. Shutdown is the default and will

occur automatically after a few seconds. Selecting Reboot will shut down and restart your system. KDE will prompt you to end a session, shutdown, or logout. (You can also open a Terminal window and enter the **shutdown**, **halt**, or **reboot** command, as described later; **halt** will log out and shut down your system.)

Switching Users

Once you have logged in to your desktop, you can switch to different user without having to log out or end your current user session. On GNOME you use the User Switcher tool, a GNOME applet on the panel. For KDE you use the Switch User entry on the Main menu.

User Switcher: GNOME

On GNOME, the switcher will appear on the panel as the name of the currently logged-in user. If you left-click the name, a list of all other users will be displayed. Check boxes next to each show which users are logged in and running. To switch a user, select the user from this menu. If the user is not already logged in, the login manager (the GDM) will appear and you can enter that user's password. If the user is already logged in, then the Login window for the lock screen will appear (you can disable the lock screen). Just enter the user's password. The user's original session will continue with the same open windows and applications running as when the user switched off. You can easily switch back and forth between logged-in users, with all users retaining their session from where they left off. When you switch off from a user, that user's running programs will continue in the background.

Right-clicking the switcher will list several user management items, such as configuring the login screen, managing users, or changing the user's password and personal information. The Preferences item lets you configure how the User Switcher is displayed on your panel. Instead of the user's name, you could use the term Users or a user icon. You can also choose whether to use a lock screen when the user switches. Disabling the lock screen option will let you switch seamlessly between logged-in users.

Switch User: KDE

On KDE, the Switch User entry on the Main menu will display a list of users you can change to. You can also elect to start a different session, hiding your current one. In effect this lets you start up your desktop again as the same user. You can also lock the current session before starting a new one. New sessions can be referenced starting with the F7 key for the first session. Use CTRL-ALT-F7 to access the first session and CTRL-ALT-F8 for the second session.

Accessing Linux from the Command Line Interface

For the command line interface, you are initially given a login prompt. The system is now running and waiting for a user to log in and use it. You can enter your username and password to use the system. The login prompt is preceded by the hostname you gave your system. In this example, the hostname is **turtle**. When you finish using Linux, you first log out. Linux then displays exactly the same login prompt, waiting for you or another user to log in again. This is the equivalent of the Login window provided by the GDM. You can then log in to another account.

```
Linux release
Kernel 2.6 on an i686

turtle login:
```

Logging In and Out with the Command Line

Once you log in to an account, you can enter and execute commands. Logging in to your Linux account involves two steps: entering your username and then entering your password. Type in the username for your user account. If you make a mistake, you can erase characters with the BACKSPACE key. In the next example, the user enters the username **richlp** and is then prompted to enter the password:

```
Linux release
Kernel 2.6 on an i686

turtle login: richlp
Password:
```

When you type in your password, it does not appear on the screen. This is to protect your password from being seen by others. If you enter either the username or the password incorrectly, the system will respond with the error message "Login incorrect" and will ask for your username again, starting the login process over. You can then reenter your username and password.

Once you enter your username and password correctly, you are logged in to the system. Your command line prompt is displayed, waiting for you to enter a command. Notice the command line prompt is a dollar sign (**$**), not a number sign (**#**). The **$** is the prompt for regular users, whereas the **#** is the prompt solely for the root user. In this version of Linux, your prompt is preceded by the hostname and the directory you are in. Both are bounded by a set of brackets.

```
[turtle /home/richlp]$
```

To end your session, issue the **logout** or **exit** command. This returns you to the login prompt, and Linux waits for another user to log in:

```
[turtle /home/richlp]$ logout
```

Shutting Down Linux from the Command Line

If you want to turn off your computer, you must first shut down Linux. Not shutting down Linux may require Linux to perform a lengthy systems check when it starts up again. You shut down your system in either of two ways. First log in to an account and then enter the **halt** command. This command will log you out and shut down the system.

```
$ halt
```

Alternatively, you can use the **shutdown** command with the **-h** option. Or, with the **-r** option, the system shuts down and then reboots. In the next example, the system is shut down after five minutes. To shut down the system immediately, you can use **+0** or the word **now**.

```
# shutdown -h now
```

TIP *Shutting down involves a series of important actions, such as unmounting file systems and shutting down any servers. You should never simply turn off the computer, though it can normally recover.*

You can also force your system to reboot at the login prompt by holding down the CTRL and ALT keys and then pressing the DEL key (CTRL-ALT-DEL). Your system will go through the standard shutdown procedure and then reboot your computer.

The GNOME and KDE Desktops

Two alternative desktop GUIs can be installed on most Linux systems: GNOME and KDE. Each has its own style and appearance. GNOME uses the Clearlooks theme for its interface with the distribution screen background and menu icon as its default.

It is important to keep in mind that though the GNOME and KDE interfaces appear similar, they are really two very different desktop interfaces with separate tools for selecting preferences. The Preferences menus on GNOME and KDE display very different selections of desktop configuration tools.

Though GNOME and KDE are wholly integrated desktops, they in fact interact with the operating system through a window manager—Metacity in the case of GNOME and the KDE window manager for KDE. You can use a different GNOME- or KDE-compliant window manager if you wish, or simply use a window manager in place of either KDE or GNOME. You can find detailed information about different window managers available for Linux from the X11 website at **xwinman.org**.

KDE

The K Desktop Environment (KDE) displays a panel at the bottom of the screen that looks very similar to one displayed on the top of the GNOME desktop. The file manager appears slightly different but operates much the same way as the GNOME file manager. There is a Control Center entry in the Main menu that opens the KDE control center, from which you can configure every aspect of KDE, such as themes, panels, peripherals like printers and keyboards, even the KDE file manager's web browsing capabilities.

NOTE *For both GNOME and KDE, the file manager is Internet-aware. You can use it to access remote FTP directories and to display or download their files, though in KDE the file manager is also a fully functional web browser.*

XFce4

The XFce4 desktop is a new lightweight desktop designed to run fast without the kind of overhead seen in full-featured desktops like KDE and GNOME. It includes its own file manager and panel, but the emphasis is on modularity and simplicity. The desktop consists of a collection of modules, including the xffm file manager, the xfce4-panel panel, and the xfwm4 window manager. In keeping with its focus on simplicity, its small scale makes it appropriate for laptops or dedicated systems that have no need for the complex overhead found in other desktops.

GNOME

The GNOME desktop display shows three menus: Applications, Places, and System. The Places menu lets you easily access commonly used locations like your home directory, the desktop folder for any files on your desktop, and the Computer window, through which you can access devices, shared file systems, and all the directories on your local system. The System menu includes Preferences and Administration menus. The Preferences menu is

used for configuring your GNOME settings, such as the theme you want to use and the behavior of your mouse.

TIP *If your desktop supports xdg-users-dirs configuration, then your home directory will already have default directories created for commonly used files. These include Download, Pictures, Documents, Music, and Videos.*

To move a window, left-click and drag its title bar. Each window supports Maximize, Minimize, and Close buttons. Double-clicking the title bar will maximize the window. Each window will have a corresponding button on the bottom panel. You can use this button to minimize and restore the window. The desktop supports full drag-and-drop capabilities. You can drag folders, icons, and applications to the desktop or other file manager windows open to other folders. The move operation is the default drag operation (you can also press the SHIFT key while dragging). To copy files, press the CTRL key and then click and drag before releasing the mouse button. To create a link, hold both the CTRL and SHIFT keys while dragging the icon to the location where you want the link, such as the desktop.

GNOME provides several tools for configuring your desktop. These are listed in the System | Preferences menu. Configuration preference tools are organized into several submenus: Personal, Look and Feel, Internet and Network, Hardware, and System. Those that do not fall into any category are listed directly. Several are discussed in different sections in this and other chapters. The Help button on each preference window will display detailed descriptions and examples. Some of the more important tools are discussed here.

The Keyboard Shortcuts configuration (Personal | Keyboard Shortcuts) lets you map keys to certain tasks, for example, mapping multimedia keys on a keyboard to media tasks such as play and pause. The File Management configuration (Personal | File Management) lets you determine the way files and directories are displayed, along with added information to show in icon captions or list views. The Windows configuration (Look and Feel | Windows) is where you can enable features like window roll-up, window movement key, and mouse window selection.

The Mouse and Keyboard preferences are the primary tools for configuring your mouse and keyboard (Hardware | Keyboard and Hardware | Mouse). The Mouse preferences let you choose a mouse image and configure its motion and hand orientation. The Keyboard preferences window shows several panels for selecting your keyboard model (Layout), configuring keys (Layout Options) and repeat delay (Keyboard), and even enforcing breaks from power typing as a health precaution.

GNOME and KDE Applets

GNOME applets are small programs that operate off your panel. It is very easy to add applets. Right-click the panel and select the Add entry. This lists all available applets. Some helpful applets are dictionary lookup; the current weather; the system monitor, which shows your CPU usage; the CPU Frequency Scaling Monitor for Cool and Quiet processors; and Search, which searches your system for files, as well as Lock, Shutdown, and Logout buttons. Some of these, including Find, Lock, and Logout, are already on the Places menu. You can drag these directly from the menu to the panel to add the applet. Following the web browser and email icons, you have, from left to right: Search for files, dictionary lookup, Tomboy note taker, Network connection monitor, CPU scaling monitor, System Monitor, Weather report,

Eyes that follow your mouse around, User Switcher, and the Logout, Shutdown, and Lock Screen buttons.

On KDE, right-click the panel and select Add Applet to Panel. From the KDE applets window, you can select similar applets such as System Monitor and Sound Mixer.

Starting a GUI from the Command Line

Once logged in to the system from the command line, you still have the option of starting an X Window System GUI, such as GNOME or KDE. In Linux, the command **startx** starts a desktop. The **startx** command starts the GNOME desktop by default. Once you shut down the desktop, you will return to your command line interface, still logged in.

```
$ startx
```

Desktop Operations

There are several desktop operations that you may want to take advantage of when first setting up your desktop. These include selecting themes, setting your font sizes larger for high resolution monitors, burning CD/DVD discs, searching your desktop for files, using removable media like USB drives, and accessing remote hosts.

Desktop Themes

On GNOME, you use the Themes Preferences tool to select or customize a theme. Themes control your desktop appearance. When you open the Theme tool, a list of currently installed themes is shown. The GNOME theme is initially selected. You can move down the list to select a different theme if you wish. If you have downloaded additional themes from sites like **art.gnome.org**, you can click the Install button to locate and install them. Once installed, the additional themes will also be displayed in the Themes Preferences tool's listing. If you downloaded and installed a theme or icon set from the Fedora repository, it will be automatically installed for you.

The true power of Themes is in the ability it provides users to customize any given theme. Themes are organized into three components: controls, window border, and icons. Controls covers the appearance of window and dialog controls such as buttons and slider bars. Window border specifies how title bars, borders, and window buttons are displayed. Icons specify how all icons used on the desktop are displayed, whether on the file manager, desktop, or the panel. You can mix and match components from any installed theme to make your own theme. You can even download and install separate components like specific icon sets, which you can then use in a customized theme.

Clicking the Customize button will open a Themes Details window with panels of the different theme components. The ones used for the current theme will be already selected. In the control, window border, and icon panels you will see listings of the different installed themes. An additional Color panel lets you set the background and text colors for windows, input boxes, and selected items. You can then mix and match different components like icons, window styles, and controls, creating your own customized theme. Upon selecting a component, your desktop automatically changes, showing you how it looks.

One you have created a new customized theme, a Custom Theme entry will appear in the theme list. To save the customized theme, click the Save Theme button. This opens

www.allitebooks.com

a dialog where you can enter a theme name, any notes, and specify whether you want to also keep the theme background. The saved theme then appears in the theme listing.

On KDE, open the Theme manager in the KDE Control Center under Appearances and Themes. Select the theme you want from the Theme panel. The selected theme will be displayed on the facing panel. Buttons in the Customize section let you build a customized theme, selecting background, icons, colors, styles, fonts, and even screensavers. To download new themes, click the Get new themes link in the upper right corner. This opens the Kde-look web page for KDE themes. You will have to download themes, extract them, and then click the Install theme button, locating and selecting the downloaded theme's **.kth** file. This method works only for themes in the Theme manager format, **kth**. Themes not in this format have to be installed manually.

GNOME themes and icons installed directly by a user are placed in the **.themes** and **.icons** directories in the user's home directory. Should you want these themes made available for all users, you can move them from the **.themes** and **.icons** directories to the **/usr/share/icons** and **/usr/share/themes** directories. Be sure to log in as the root user. You then need to change ownership of the moved themes and icons to the root user:

```
chown -R root:root  /usr/share/themes/newtheme
```

User KDE themes are placed in the **.kde/share/apps/kthememanager** directory.

Fonts

Most distributions now use the fontconfig method for managing fonts (**fontconfig.org**). You can easily change font sizes, add new fonts, and configure features like anti-aliasing. Both GNOME and KDE provide tools for selecting, resizing, and adding fonts.

Resizing Desktop Fonts

With very large monitors and their high resolutions becoming more common, one feature users find helpful is the ability to increase the desktop font sizes. On a large widescreen monitor, resolutions less than the native one tend not to scale well. A monitor always looks best in its native resolution. However, with a large native resolution like 1900 × 1200, text sizes become so small they are hard to read. You can overcome this issue by increasing the font size. Use the font tools on your desktop to change these sizes (System | Preferences | Look And Feel | Fonts on GNOME; for KDE, select the Fonts entry in the Control Center's Appearance and Themes).

Adding Fonts

To add a new font (for both GNOME and KDE), just enter the **fonts:/** URL in a file manager window (Open Location in the GNOME File menu). This opens the font window. Drag and drop your font file to it. When you restart, your font will be available for use on your desktop. KDE will have Personal and System folders for fonts, initially showing icons for each. For user fonts, open the Personal Fonts window. Fonts that are Zip archived, should first be opened with the Archive manager and then can be dragged from the archive manager to the font viewer. To remove a font, right-click it in the font viewer and select Move to Trash or Delete.

User fonts will be installed to a user's **.fonts** directory. For fonts to be available to all users, they have to be installed in the **/usr/share/fonts** directory, making them system fonts. On KDE, you do this by opening the System folder, instead of the Personal folder, when you start up the fonts viewer. You can do this from any user login. Then drag any font packages

to this **fonts:/System** window. On GNOME, you have to log in as the root user and manually copy fonts to the **/usr/share/fonts** directory. If your system has both GNOME and KDE installed, you can install system fonts using KDE (Konqueror file manager), and they will be available on GNOME.

To provide speedy access to system fonts, you should create font information cache files for the **/usr/share/fonts** directory. To do this, run the **fc-cache** command as the root user.

Configuring Fonts

On GNOME, to better refine your font display, you can use the font rendering tool. Open the Font Preferences tool (System | Preferences | Look and Feel | Fonts). In the Font Rendering section are basic font rendering features like Monochrome, Best contrast, Best shapes, and Subpixel smoothing. Choose the one that works best. For LCDS, choose Subpixel smoothing. For detailed configuration, click the Details button. Here you can set smoothing, hinting (anti-aliasing), and subpixel color order features. The subpixel color order is hardware dependent. On KDE, in the KDE control center, select the Fonts entry under Appearance and Themes. Click the Use anti-aliasing for fonts check box, and then click the Configure button to open a window to let you select hinting and subpixel options.

On GNOME, clicking a font entry in the Fonts Preferences tool will open a Pick a Font dialog that will list all available fonts. On KDE, clicking any of the Choose buttons on the Control Center's Fonts panel will also open a window listing all available fonts. You can also generate a listing with the **fc-list** command. The list will be unsorted, so you should pipe it first to the sort command. You can use **fc-list** with any font name or name pattern to search for fonts, with options to search by language, family, or styles. See the **/etc/share/fontconfig** documentation for more details.

```
fc-list | sort
```

TIP *Microsoft common web fonts are freely available from* **fontconfig.org**. *These fonts are archived in Microsoft's cab format. You will need to download and install the* **cabextract** *tool (available from most distribution software collections and repositories) to extract the fonts. Once extracted, you can copy them to a folder in the* **/usr/share/fonts** *directory to make them available to all users. If you have access to a Windows system, you can also directly copy fonts from the Windows fonts directory to your* **/usr/share/fonts** *directory.*

Configuring Your Personal Information

On GNOME, the About Me preferences dialog lets you set up personal information to be used with your desktop applications, as well as change your password. Clicking the Image icon in the top left corner opens a browser window where you can select the image to use. The Faces directory is selected by default with images you can use. The selected image is displayed to the right in the browser window. For a personal photograph, you can use the Pictures folder. This is the Pictures folder in your home directory. Should you place a photograph or image there, you can then select if for your personal image. The image will be used in the Login screen when showing your user entry. Should you want to change your password, you can click the Change password button at the top right.

There are three panels: Contact, Address, and Personal Info. On the Contact panel you enter email (home and work), telephone, and instant messaging addresses. On the Address panel you enter your home and work addresses, and on the Personal Info panel you list your web addresses and work information.

On KDE, you can select the Password panel in the Security entry on the KDE Control Center. Here you can select a picture for your account. Contact information is handled by other applications, like Kontact for mail and user information.

Sessions

You can configure your desktop to restore your previously opened windows and applications, as well as specify startup programs. When you log out, you may want the windows you have open and the applications you have running to be automatically started when you log back in. In effect, you are saving your current session and having it restored it when you log back in. For example, if you are in the middle of working on a spreadsheet, you can save your work but not close the file. Then log out. When you log back in, your spreadsheet will be opened automatically to where you left off.

For GNOME, saving sessions is not turned on by default. You use the Sessions preferences dialog's Session Options panel (System | Preferences | Personal | Sessions) to save sessions. You can save your current session manually or opt to have all your sessions saved automatically when you log out, restoring them whenever you log in.

On KDE you can configure your session manager by selecting Sessions from the KDE Components entry in the Control Center. By default, the previous session is restored when you log in. You can also determine default shutdown behavior.

Using Removable Devices and Media

Linux desktops now support removable devices and media such as digital cameras, PDAs, card readers, and even USB printers. These devices are handled automatically with an appropriate device interface set up on the fly when needed. Such hotplugged devices are identified, and where appropriate, their icons will appear in the file manager window. For example, when you connect a USB drive to your system, it will be detected and displayed as storage device with its own file system.

Tip *When you insert a blank DVD or CD, a window will open labeled CD/DVD Creator. Burning data to a DVD or CD is a simple matter of dragging files to that window and clicking the Write To Disc button.*

Installing Multimedia Support: MP3, DVD, and DivX

Because of licensing and other restrictions, many Linux distributions do not include MP3, DVD, or DivX media support in their free versions. You have to purchase their commercial versions, which include the appropriate licenses for this support. Alternatively, you can obtain this support from independent operations such as those at **fluendo.com**. DivX support can be obtained from **labs.divx.com/DivXLinuxCodec**. Check the multimedia information pages at your distribution website for more information.

Command Line Interface

When using the command line interface, you are given a simple prompt at which you type in your command. Even with a GUI, you sometimes need to execute commands on a command line. The Terminal window is no longer available on the GNOME desktop menu. You now have to access it from the Applications | System Tools menu. If you use Terminal

windows frequently, you may want to just drag the menu entry to the desktop to create a desktop icon for the Terminal window. Just click to open.

Linux commands make extensive use of options and arguments. Be careful to place your arguments and options in their correct order on the command line. The format for a Linux command is the command name followed by options, and then by arguments, as shown here:

```
$ command-name options arguments
```

An *option* is a one-letter code preceded by one or two hyphens, which modifies the type of action the command takes. Options and arguments may or may not be optional, depending on the command. For example, the **ls** command can take an option, **-s**. The **ls** command displays a listing of files in your directory, and the **-s** option adds the size of each file in blocks. You enter the command and its option on the command line as follows:

```
$ ls -s
```

An *argument* is data the command may need to execute its task. In many cases, this is a filename. An argument is entered as a word on the command line after any options. For example, to display the contents of a file, you can use the **more** command with the file's name as its argument. The **less** or **more** command used with the filename **mydata** would be entered on the command line as follows:

```
$ less mydata
```

The command line is actually a buffer of text you can edit. Before you press ENTER, you can perform editing commands on the existing text. The editing capabilities provide a way to correct mistakes you may make when typing in a command and its options. The BACKSPACE and DEL keys let you erase the character you just typed in. With this character-erasing capability, you can BACKSPACE over the entire line if you want, erasing what you entered. CTRL-U erases the whole line and enables you to start over again at the prompt.

TIP *You can use the UP ARROW key to redisplay your last-executed command. You can then reexecute that command, or you can edit it and execute the modified command. This is helpful when you have to repeat certain operations over and over, such as editing the same file. This is also helpful when you've already executed a command you entered incorrectly.*

Help Resources

A great deal of support documentation is already installed on your system and is also accessible from online sources. Table 2-1 lists Help tools and resources accessible on most Linux systems. Both the GNOME and KDE desktops feature Help systems that use a browser-like interface to display help files. To start the GNOME or KDE Help browser, select the Help entry in the main menu. You can then choose from the respective desktop user guides, including the KDE manual, Linux Man pages, and GNU info pages. The GNOME Help Browser also accesses documents for GNOME applications such as the File Roller archive tool and Evolution mail client. The GNOME Help browser and the KDE Help Center also incorporate browser capabilities, including bookmarks and history lists for documents you view.

Resource	Description
KDE Help Center	KDE Help tool, GUI for documentation on KDE desktop and applications, Man pages, and info documents
GNOME Help Browser	GNOME Help tool, GUI for accessing documentation for the GNOME desktop and applications, Man pages, and info documents
/usr/share/doc	Location of application documentation
man *command*	Linux Man pages, detailed information on Linux commands, including syntax and options
info *application*	GNU info pages, documentation on GNU applications

TABLE 2-1 Information Resources

Context-Sensitive Help

Both GNOME and KDE, along with applications, provide context-sensitive help. Each KDE and GNOME application features detailed manuals that are displayed using their respective Help browsers. Also, system administrative tools feature detailed explanations for each task.

Application Documentation

On your system, the **/usr/share/doc** directory contains documentation files installed by each application. Within each directory, you can usually find HOW-TO, README, and INSTALL documents for that application.

The Man Pages

You can also access the Man pages, which are manuals for Linux commands available from the command line interface, using the **man** command. Enter **man** with the command for which you want information. The following example asks for information on the **ls** command:

```
$ man ls
```

Pressing the SPACEBAR key advances you to the next page. Pressing the B key moves you back a page. When you finish, press the Q key to quit the Man utility and return to the command line. You activate a search by pressing either the slash (/) or question mark (?). The / searches forward; the ? searches backward. When you press the /, a line opens at the bottom of your screen, and you then enter a word to search for. Press ENTER to activate the search. You can repeat the same search by pressing the N key. You needn't reenter the pattern.

TIP *You can also use either the GNOME or KDE Help system to display Man and info pages.*

The Info Pages

Online documentation for GNU applications, such as the GNU C and C++ compiler (gcc) and the Emacs editor, also exist as *info* pages accessible from the GNOME and KDE Help Centers. You can also access this documentation by entering the command **info.** This brings up a special screen listing different GNU applications. The info interface has its own

set of commands. You can learn more about it by entering **`info info`**. Typing **`m`** opens a line at the bottom of the screen where you can enter the first few letters of the application. Pressing ENTER brings up the info file on that application.

Software Repositories

For most Linux distributions, software has grown so large and undergoes such frequent updates that it no longer makes sense to use discs as the primary means of distribution. Instead, distribution is effected using an online software repository. This repository contains an extensive collection of distribution-compliant software.

This entire approach heralds a move from thinking of most Linux software as something included on a few discs to viewing the disc as a core from which you can expand your installed software as you like from online repositories. Most software is now located at the Internet-connected repositories. You can now think of that software as an easily installed extension of your current collection. Relying on disc media for your software has become, in a sense, obsolete.

Windows Access and Applications

In many cases, certain accommodations need to be made for Windows systems. Most Linux systems are part of networks that also run Windows systems. Using Linux Samba servers, your Linux and Windows systems can share directories and printers. In addition, you may also need to run a Windows applications directly on your Linux system. Though there is an enormous amount of Linux software available, in some cases you may need or prefer to run a Windows application. The Wine compatibility layer allows you to do just that for many Windows applications (but not all).

Setting Up Windows Network Access: Samba

Most local and home networks may include some systems working on Microsoft Windows and others on Linux. You may need to let a Windows computer access a Linux system or vice versa. Windows, because of its massive market presence, tends to benefit from both drivers and applications support not found for Linux. Though there are equivalent applications on Linux, many of which are as good or better, some applications run best on Windows, if for no other reason than that the vendor only develops drivers for Windows.

One solution is to use the superior server and storage capabilities of Linux to manage and hold data, while using Windows systems with their unique applications and drivers to run applications. For example, you can use a Linux system to hold pictures and videos, while using Windows systems to show or run them. Video or pictures can be streamed through your router to the system that wants to run them. In fact, many commercial DVR systems use a version of Linux to manage video recording and storage. Another use would be to enable Windows systems to use devices like printers that may be connected to a Linux system, or vice versa.

To allow Windows to access a Linux system and Linux to access a Windows system, you use the Samba server. Samba has two methods of authentication, shares and users, though the shares method has been deprecated. User authentication requires that there be corresponding accounts in the Windows and Linux systems. You need to set up a Samba

user with a Samba password. The Samba user should be the same name as an established account. The Windows user and Samba user can have the same name, though a Windows user can be mapped to a Samba user. A share can be made open to specific users and function as an extension of the user's storage space. On most current distributions, Samba user and password information are kept in tdb (trivial data base) Samba database files, which can be edited and added to using the **pdbedit** command.

To set up simple file sharing on a Linux system, you first need to configure your Samba server. You can do this by directly editing the **/etc/samba/samba.conf** file. If you just edit the **/etc/samba/samba.conf** file, you first need to specify the name of your Windows network. Samba provides a configuration tool called SWAT that you can use with any browser to configurae your Samba server, adding users and setting up shares. Some distributions, like Ubuntu, set up Samba automatically. KDE also provides Samba configuration.

Once set up, both GNOME and KDE allow you to browse and access Samba shares from your desktop, letting you also access shared Windows directories and printers on other systems. On GNOME click the Network and then the Windows Network icon on the My Computer window. You will see an icon for your Windows network. On either GNOME or KDE you can enter the **smb:** URL in the a file manager window to access your Windows networks.

When a Windows user wants to access the share on the Linux system, they open their My Network Places (Network on Vista) and then select Add a network place to add a network place entry for the share, or View workgroup computers to see computers on your Windows network. Selecting the Linux Samba server will display your Samba shares. To access the share, the user will be required to enter the Samba username and the Samba password. You have the option of having the username and password remembered for automatic access.

NOTE *The Fuse-smb tool lets you browse your entire Windows network at once.*

Running Windows Software on Linux: Wine

Wine is a Windows compatibility layer that will allow you to run many Windows applications natively on Linux. Though you could run the Windows operating system on it, the actual Windows operating system is not required. Windows applications will run as if they were Linux applications, able to access the entire Linux file system and use Linux-connected devices. Applications that are heavily driver dependent, such as graphic-intensive games, most likely will not run. Others, such as newsreaders, which do not rely on any specialized drivers, may run very well. For some applications, you may also need to copy over specific Windows DLLs from a working Windows system to your Wine Windows **system32** or **system** directory.

To install Wine on your system, search for **wine** on you distributions repositories. For some distributions you may have to download wine directly from **winehq.org.** Binaries for several distributions are provided.

TIP *To play Windows games on Linux, you can try using cedega. These are inexpensive commercial drivers that are configured to support many popular games,* **cedega.com**, *enabling full graphics acceleration.*

Once installed, a Wine menu will appear in the Applications menu. The Wine menu holds entries for Wine configuration, the Wine software uninstaller, and the Wine file browser, as well as a regedit registry editor, a notepad, and a Wine help tool.

To set up Wine, a user starts the Wine Configuration tool. This opens a window with panels for Applications, Libraries (DLL selection), Audio (sound drivers), Drives, Desktop Integration, and Graphics. On the Applications panel you can select which version of Windows an application is designed for. The Drives panel will list your detected partitions, as well as your Windows-emulated drives, such as drive C:. The C: drive is really just a directory, **.wine/drive_c**, not a partition of a fixed size. Your actual Linux file system will be listed as the Z: drive.

Once configured, Wine will set up a **.wine** directory on the user's home directory (the directory is hidden, so enable Show Hidden Files in the file browser View menu to display it). Within that directory will be the **drive_c** directory, which functions as the C: drive, holding your Windows system files and program files in the **Window**s and **Program File** subdirectories. The **System and System32** directories are located in the **Windows** directory. Here is where you place any needed DLL files. The **Program Files** directory will hold your installed Windows programs, just as they would be installed on a Windows **Program Files** directory.

To install a Windows application with Wine, you can either use the Wine configuration tool or open a Terminal window and run the **`wine`** command with the Windows application as an argument. The following example installs the popular newsbin program:

```
$ wine newsbin.exe
```

To install with the Windows Configuration tool, select the Applications panel and then click Add.

Some applications, such as newsbin, will also require that you use certain DLL files from a working Windows operating system. The DLL files are normally copied to the user's **.wine/drive_c/Windows/system32** directory.

Icons for installed Windows software will appear on your desktop. Just double-click an icon to start up the application. It will run normally within a Linux window, as would any Linux application.

Installing Windows fonts on Wine is a simple matter of copying fonts from a Windows font directory to your Wine **.wine/drive_c/Windows/fonts** directory. You can just copy any Windows **.ttf** file to this directory to install a font. You can also use the Microsoft common web fonts available from **fontconfig.org** (this will require **cabextract** to extract them).

Wine will use a stripped-down window style for features like buttons and the title bar. If you want to use the XP style, download and install the Royal theme from Microsoft. Keep in mind, however, that supporting this theme is very resource intensive and will likely slow down your system.

TIP *Alternatively, you can use the commercial Windows compatibility layer called CrossoverOffice. This is a commercial product tested to run certain applications like Microsoft Office. Check* **codeweavers.com** *for more details. CrossoverOffice is based on Wine, which CodeWeavers supports directly.*

PART II

The Linux Shell and File Structure

CHAPTER 3
The Shell

CHAPTER 4
The Shell Scripts and Programming

CHAPTER 5
Shell Configuration

CHAPTER 6
Linux Files, Directories, and Archives

This page intentionally left blank

www.allitebooks.com

CHAPTER 3
The Shell

The *shell* is a command interpreter that provides a line-oriented interactive and noninteractive interface between the user and the operating system. You enter commands on a command line; they are interpreted by the shell and then sent as instructions to the operating system (the command line interface is accessible from GNOME and KDE through a Terminal windows—Applications / Accessories menu). You can also place commands in a script file to be consecutively executed, much like a program. This interpretive capability of the shell provides for many sophisticated features. For example, the shell has a set of file matching characters that can generate filenames. The shell can redirect input and output, as well as run operations in the background, freeing you to perform other tasks.

Several different types of shells have been developed for Linux: the Bourne Again shell (BASH), the Korn shell, the TCSH shell, and the Z shell. TCSH is an enhanced version of the C shell used on many Unix systems, especially BSD versions. You need only one type of shell to do your work. Linux includes all the major shells, although it installs and uses the BASH shell as the default. If you use the command line shell, you will be using the BASH shell unless you specify another. This chapter primarily discusses the BASH shell, which shares many of the same features as other shells. A brief discussion of the C shell, TCSH, and the Z shell follows at the end of the chapter, noting differences.

You can find out more about shells at their respective websites, as listed in Table 3-1. Also, a detailed online manual is available for each installed shell. Use the **man** command and the shell's keyword to access them, **bash** for the BASH shell, **ksh** for the Korn shell, **zsh** for the Z shell, and **tsch** for the TSCH shell. For the C shell you can use **csh**, which links to **tcsh**. For example, the command **man bash** will access the BASH shell online manual.

NOTE *You can find out more about the BASH shell at* ***gnu.org/software/bash.*** *A detailed online manual is available on your Linux system using the* **man** *command with the* **bash** *keyword.*

The Command Line

The Linux command line interface consists of a single line into which you enter commands with any of their options and arguments. From GNOME or KDE, you can access the command line interface by opening a terminal window. Should you start Linux with the command line interface, you will be presented with a BASH shell command line when you log in.

URL	Shell
gnu.org/software/bash	BASH website with online manual, FAQ, and current releases
gnu.org/software/bash/manual/bash.html	BASH online manual
zsh.org	Z shell website with referrals to FAQs and current downloads
tcsh.org	TCSH website with detailed support including manual, tips, FAQ, and recent releases
kornshell.com	Korn shell site with manual, FAQ, and references

TABLE 3-1 Linux Shells Websites

By default, the BASH shell has a dollar sign (**$**) prompt, but Linux has several other types of shells, each with its own prompt (% for the C shell, for example). The root user will have a different prompt, the #. A shell *prompt,* such as the one shown here, marks the beginning of the command line:

```
$
```

You can enter a command along with options and arguments at the prompt. For example, with an **-l** option, the **ls** command will display a line of information about each file, listing such data as its size and the date and time it was last modified. The dash before the **-l** option is required. Linux uses it to distinguish an option from an argument.

```
$ ls -l
```

If you want the information displayed only for a particular file, you can add that file's name as the argument, following the **-l** option:

```
$ ls -l mydata
-rw-r--r-- 1 chris weather 207 Feb 20 11:55 mydata
```

TIP *Some commands can be complex and take some time to execute. When you mistakenly execute the wrong command, you can interrupt and stop such commands with the interrupt key—CTRL-C.*

You can enter a command on several lines by typing a backslash just before you press ENTER. The backslash "escapes" the ENTER key, effectively continuing the same command line to the next line. In the next example, the **cp** command is entered on three lines:

```
$ cp -i \
mydata \
/home/george/myproject/newdata
```

You can also enter several commands on the same line by separating them with a semicolon (**;**). In effect the semicolon operates as an execute operation. Commands will be executed in the sequence they are entered. The following command executes an **ls** command followed by a **date** command.

```
$ ls ; date
```

You can also conditionally run several commands on the same line with the **&&** operator (see Chapter 4). A command is executed only if the previous one is true. This feature is useful for running several dependent scripts on the same line. In the next example, the **ls** command runs only if the **date** command is successfully executed.

```
$ date && ls
```

TIP *Commands can also be run as arguments on a command line, using their results for other commands. To run a command within a command line, you encase the command in back quotes; see "Values from Linux Commands" in chapter 4.*

Command Line Editing

The BASH shell, which is your default shell, has special command line editing capabilities that you may find helpful as you learn Linux (see Table 3-2). You can easily modify commands you have entered before executing them, moving anywhere on the command line and inserting or deleting characters. This is particularly helpful for complex commands. You can use the CTRL-F or RIGHT ARROW key to move forward a character or the CTRL-B or LEFT ARROW key to move back a character. CTRL-D or DEL deletes the character the cursor is on, and CTRL-H or BACKSPACE deletes the character before the cursor. To add text, you use the arrow keys to move the cursor to where you want to insert text and type the new characters. You can even cut words with the CTRL-W or ALT-D key and then use the CTRL-Y key to paste them back in at a different position, effectively moving the words. As a rule, the CTRL version of the command operates on characters, and the ALT version works on words, such as CTRL-T to transpose characters and ALT-T to transpose words. At any time, you can press ENTER to execute the command. The actual associations of keys and their tasks, along with global settings, are specified in the **/etc/inputrc** file.

TIP *The editing capabilities of the BASH shell command line are provided by Readline, which supports numerous editing operations. You can even bind a key to a selected editing operation. Readline uses the **/etc/inputrc** file to configure key bindings. This file is read automatically by your **/etc/profile** shell configuration file when you log in (see Chapter 5). Users can customize their editing commands by creating an **.inputrc** file in their home directory (this is a dot file). It may be best to first copy the **/etc/inputrc** file as your **.inputrc** file and then edit it. **/etc/profile** will first check for a local **.inputrc** file before accessing the **/etc/inputrc** file. You can find out more about Readline in the BASH shell reference manual at **gnu.org/manual/bash**.*

Movement Commands	Operation
CTRL-F, RIGHT-ARROW	Move forward a character.
CTRL-B, LEFT-ARROW	Move backward a character.
CTRL-A or HOME	Move to beginning of line.
CTRL-E or END	Move to end of line.
ALT-F	Move forward a word.
ALT-B	Move backward a word.
CTRL-L	Clear screen and place line at top.
Editing Commands	**Operation**
CTRL-D or DEL	Delete character cursor is on.
CTRL-H or BACKSPACE	Delete character before the cursor.
CTRL-K	Cut remainder of line from cursor position.
CTRL-U	Cut from cursor position to beginning of line.
CTRL-W	Cut previous word.
CTRL-C	Cut entire line.
ALT-D	Cut the remainder of a word.
ALT-DEL	Cut from the cursor to the beginning of a word.
CTRL-Y	Paste previous cut text.
ALT-Y	Paste from set of previously cut text.
CTRL-Y	Paste previous cut text.
CTRL-V	Insert quoted text, used for inserting control or meta (ALT) keys as text, such as CTRL-B for backspace or CTRL-T for tabs.
ALT-T	Transpose current and previous word.
ALT-L	Lowercase current word.
ALT-U	Uppercase current word.
ALT-C	Capitalize current word.
CTRL-SHIFT-_	Undo previous change.

TABLE 3-2 Command Line Editing Operations

Command and Filename Completion

The BASH command line has a built-in feature that performs command line and filename completion. Automatic completions can be effected using the TAB key. If you enter an incomplete pattern as a command or filename argument, you can then press the TAB key to activate the command and filename completion feature, which completes the pattern.

Directories will have a / attached to their name. If more than one command or file has the same prefix, the shell simply beeps and waits for you to enter the TAB key again. It then displays a list of possible command completions and waits for you to add enough characters to select a unique command or filename. In situations where you know there are likely multiple possibilities, you can just press the ESC key instead of two TABs. In the next example, the user issues a **cat** command with an incomplete filename. When the user presses the TAB key, the system searches for a match and, when it finds one, fills in the filename. The user can then press ENTER to execute the command.

```
$ cat pre tab
$ cat preface
```


Automatic completion also works with the names of variables, users, and hosts. In this case, the partial text needs to be preceded by a special character indicating the type of name. Variables begin with a $ sign, so any text beginning with a $ sign is treated as a variable to be completed. Variables are selected from previously defined variables, like system shell variables (see Chapter 4). Usernames begin with a tilde (~). Host names begin with an @ sign, with possible names taken from the **/etc/hosts** file. A listing of possible automatic completions follows:

- Filenames begin with any text or /.
- Shell variable text begins with a $ sign.
- Username text begins with a ~ sign.
- Host name text begins with a @.
- Commands, aliases, and text in files begin with normal text.

For example, to complete the variable HOME given just $HOM, simply enter a TAB character.

```
$ echo $HOM <tab>
$ echo $HOME
```

If you enter just an H, then you can enter two tabs to see all possible variables beginning with H. The command line will be redisplayed, letting you complete the name.

```
$ echo $H <tab> <tab>
$HISTCMD $HISTFILE $HOME $HOSTTYPE HISTFILE  $HISTSIZE $HISTNAME
$ echo $H
```

You can also specifically select the kind of text to complete, using corresponding command keys. In this case, it does not matter what kind of sign a name begins with. For example, the ALT-~ will treat the current text as a username. ALT-@ will treat it as a host name and ALT-$, as a variable. ALT-! will treat it as a command. To display a list of possible completions, use the CTRL-X key with the appropriate completion key, as in CTRL-X-$ to list possible variable completions. See Table 3-3 for a complete listing.

Command (CTRL-R for Listing Possible Completions)	Description
TAB	Automatic completion
TAB TAB or ESC	List possible completions
ALT-/, CTRL-R-/	Filename completion, normal text for automatic
ALT-$, CTRL-R-$	Shell variable completion, $ for automatic
ALT-~, CTRL-R-~	Username completion, ~ for automatic
ALT-@, CTRL-R-@	Host name completion, @ for automatic
ALT-!, CTRL-R-!	Command name completion, normal text for automatic

TABLE 3-3 Command Line Text Completion Commands

History

The BASH shell keeps a list, called a *history list,* of your previously entered commands. You can display each command, in turn, on your command line by pressing the UP ARROW key. The DOWN ARROW key moves you down the list. You can modify and execute any of these previous commands when you display them on your command line.

TIP *The capability to redisplay a previous command is helpful when you've already executed a command you entered incorrectly. In this case, you are presented with an error message and a new, empty command line. By pressing the up arrow key, you can redisplay your previous command, make corrections to it, and then execute it again. This way, you do not have to enter the whole command again.*

History Events

In the BASH shell, the *history utility* keeps a record of the most recent commands you have executed. The commands are numbered starting at 1, and a limit exists to the number of commands remembered—the default is 500. The history utility is a kind of short-term memory, keeping track of the most recent commands you have executed. To see the set of your most recent commands, type **history** on the command line and press ENTER. A list of your most recent commands is then displayed, preceded by a number.

```
$ history
1 cp mydata today
2 vi mydata
3 mv mydata reports
4 cd reports
5 ls
```

Each of these commands is technically referred to as an event. An *event* describes an action that has been taken—a command that has been executed. The events are numbered according to their sequence of execution. The most recent event has the largest number. Each of these events can be identified by its number or beginning characters in the command.

The history utility enables you to reference a former event, placing it on your command line and enabling you to execute it. The easiest way to do this is to use the UP ARROW and DOWN ARROW keys to place history events on your command line, one at a time. You needn't display the list first with **history**. Pressing the UP ARROW key once places the last history event on your command line. Pressing it again places the next history event on your command. Pressing the DOWN ARROW key places the next event on the command line.

You can use certain control and meta keys to perform other history operations like searching the history list. A meta key is the ALT key, or the ESC key on keyboards that have no ALT key. The ALT key is used here. ALT-< will move you to the beginning of the history list; ALT-N will search it. CTRL-S and CTRL-R will perform incremental searches, displaying matching commands as you type in a search string. Table 3-4 lists the different commands for referencing the history list.

TIP *If more than one history event matches what you have entered, you will hear a beep, and you can then enter more characters to help uniquely identify the event.*

History Commands	Description
CTRL-N or DOWN ARROW	Move down to the next event in the history list.
CTRL-P or UP ARROW	Move up to the previous event in the history list.
ALT-<	Move to the beginning of the history event list.
ALT->	Move to the end of the history event list.
ALT-N	Forward search, next matching item.
ALT-P	Backward search, previous matching item.
CTRL-S	Forward search history, forward incremental search.
CTRL-R	Reverse search history, reverse incremental search.
fc *event-reference*	Edits an event with the standard editor and then executes it **Options** **-l** List recent history events; same as **history** command **-e** *editor event-reference*; invokes a specified editor to edit a specific event
History Event References	
!*event num*	References an event by its event number.
!!	References the previous command.
!*characters*	References an event beginning with the specified characters.
!?*pattern***?**	References an event containing the specified pattern.
!-*event num*	References an event with an offset from the first event.
!*num***-***num*	References a range of events.

TABLE 3-4 History Commands and History Event References

You can also reference and execute history events using the **!** history command. The **!** is followed by a reference that identifies the command. The reference can be either the number of the event or a beginning set of characters in the event. In the next example, the third command in the history list is referenced first by number and then by the beginning characters:

```
$ !3
mv mydata reports
$ !mv my
mv mydata reports
```

You can also reference an event using an offset from the end of the list. A negative number will offset from the end of the list to that event, thereby referencing it. In the next example, the fourth command, **cd mydata**, is referenced using a negative offset, and then executed. Remember that you are offsetting from the end of the list—in this case, event 5—up toward the beginning of the list, event 1. An offset of 4 beginning from event 5 places you at event 2.

```
$ !-4
vi mydata
```

To reference the last event, you use a following !, as in **!!**. In the next example, the command **!!** executes the last command the user executed—in this case, **ls**:

```
$ !!
ls
mydata today reports
```

History Event Editing

You can also edit any event in the history list before you execute it. In the BASH shell, you can do this two ways. You can use the command line editor capability to reference and edit any event in the history list. You can also use a history **fc** command option to reference an event and edit it with the full Vi editor. Each approach involves two different editing capabilities. The first is limited to the commands in the command line editor, which edits only a single line with a subset of Emacs commands. At the same time, however, it enables you to reference events easily in the history list. The second approach invokes the standard Vi editor with all its features, but only for a specified history event.

With the command line editor, not only can you edit the current command, you can also move to a previous event in the history list to edit and execute it. The CTRL-P command then moves you up to the prior event in the list. The CTRL-N command moves you down the list. The ALT-< command moves you to the top of the list, and the ALT-> command moves you to the bottom. You can even use a pattern to search for a given event. The slash followed by a pattern searches backward in the list, and the question mark followed by a pattern searches forward in the list. The **n** command repeats the search.

Once you locate the event you want to edit, you use the Emacs command line editing commands to edit the line. CTRL-D deletes a character. CTRL-F or the RIGHT ARROW moves you forward a character, and CTRL-B or the LEFT ARROW moves you back a character. To add text, you position your cursor and type in the characters you want.

If you want to edit an event using a standard editor instead, you need to reference the event using the **fc** command and a specific event reference, such as an event number.

The editor used is the one specified by the shell in the **FCDIT** or **EDITOR** variable. This serves as the default editor for the **fc** command. You can assign to the **FCDIT** or **EDITOR** variable a different editor if you wish, such as Emacs instead of Vi. The next example will edit the fourth event, **cd reports**, with the standard editor and then execute the edited event:

```
$ fc 4
```

You can select more than one command at a time to be edited and executed by referencing a range of commands. You select a range of commands by indicating an identifier for the first command followed by an identifier for the last command in the range. An identifier can be the command number or the beginning characters in the command. In the next example, the range of commands 2 through 4 is edited and executed, first using event numbers and then using beginning characters in those events:

```
$ fc 2 4
$ fc vi c
```

The **fc** command uses the default editor specified in the **FCEDIT** special variable (**If FCEDIT** is not defined, it checks for the **EDITOR** variable. If neither is defined it uses Vi). Usually, this is the Vi editor. If you want to use the Emacs editor instead, you use the **-e** option and the term **emacs** when you invoke **fc**. The next example will edit the fourth event, **cd reports**, with the Emacs editor and then execute the edited event:

```
$ fc -e emacs 4
```

Configuring History: HISTFILE and HISTSAVE

The number of events saved by your system is kept in a special system variable called **HISTSIZE**. By default, this is usually set to 500. You can change this to another number by simply assigning a new value to **HISTSIZE**. In the next example, the user changes the number of history events saved to 10:

```
$ HISTSIZE=10
```

The actual history events are saved in a file whose name is held in a special variable called **HISTFILE**. By default, this file is the **.bash_history** file. You can change the file in which history events are saved, however, by assigning its name to the **HISTFILE** variable. In the next example, the value of **HISTFILE** is displayed. Then a new filename is assigned to it, **newhist**. History events are then saved in the **newhist** file.

```
$ echo $HISTFILE
.bash_history
$ HISTFILE="newhist"
$ echo $HISTFILE
newhist
```

Filename Expansion: *, ?, [ ]

Filenames are the most common arguments used in a command. Often you may know only part of the filename, or you may want to reference several filenames that have the same extension or begin with the same characters. The shell provides a set of special characters

that search out, match, and generate a list of filenames. These are the asterisk, the question mark, and brackets (`*`, `?`, `[]`). Given a partial filename, the shell uses these matching operators to search for files and expand to a list of filenames found. The shell replaces the partial filename argument with the expanded list of matched filenames. These filenames can then become the arguments for commands such as `ls`, which can operate on many files. Table 3-5 lists the shell's file expansion characters.

Common Shell Symbols	Execution
ENTER	Execute a command line.
;	Separate commands on the same command line.
`` `command` ``	Execute a command.
`$(command)`	Execute a command.
`[]`	Match on a class of possible characters in filenames.
\	Quote the following character. Used to quote special characters.
|	Pipe the standard output of one command as input for another command.
`&`	Execute a command in the background.
`!`	History command.
File Expansion Symbols	**Execution**
`*`	Match on any set of characters in filenames.
`?`	Match on any single character in filenames.
`[]`	Match on a class of characters in filenames.
Redirection Symbols	**Execution**
`>`	Redirect the standard output to a file or device, creating the file if it does not exist and overwriting the file if it does exist.
`>!`	Force the overwriting of a file if it already exists. This overrides the `noclobber` option.
`<`	Redirect the standard input from a file or device to a program.
`>>`	Redirect the standard output to a file or device, appending the output to the end of the file.
Standard Error Redirection Symbols	**Execution**
`2>`	Redirect the standard error to a file or device.
`2>>`	Redirect and append the standard error to a file or device.
`2>&1`	Redirect the standard error to the standard output.
`>&`	Redirect the standard error to a file or device.
`|&`	Pipe the standard error as input to another command.

TABLE 3-5 Shell Symbols

Matching Multiple Characters

The asterisk (*****) references files beginning or ending with a specific set of characters. You place the asterisk before or after a set of characters that form a pattern to be searched for in filenames. If the asterisk is placed before the pattern, filenames that end in that pattern are searched for. If the asterisk is placed after the pattern, filenames that begin with that pattern are searched for. Any matching filename is copied into a list of filenames generated by this operation. In the next example, all filenames beginning with the pattern "doc" are searched for and a list is generated. Then all filenames ending with the pattern "day" are searched for and a list is generated. The last example shows how the ***** can be used in any combination of characters.

```
$ ls
doc1 doc2 document docs mydoc monday tuesday
$ ls doc*
doc1 doc2 document docs
$ ls *day
monday tuesday
$ ls m*d*
monday
$
```

Filenames often include an extension specified with a period and followed by a string denoting the file type, such as **.c** for C files, **.cpp** for C++ files, or even **.jpg** for JPEG image files. The extension has no special status and is only part of the characters making up the filename. Using the asterisk makes it easy to select files with a given extension. In the next example, the asterisk is used to list only those files with a **.c** extension. The asterisk placed before the **.c** constitutes the argument for **ls**.

```
$ ls *.c
calc.c main.c
```

You can use ***** with the **rm** command to erase several files at once. The asterisk first selects a list of files with a given extension or beginning or ending with a given set of characters and then it presents this list of files to the **rm** command to be erased. In the next example, the **rm** command erases all files beginning with the pattern "doc":

```
$ rm doc*
```

TIP *Use the* ***** *file expansion character carefully and sparingly with the* **rm** *command. The combination can be dangerous. A misplaced* ***** *in an* **rm** *command without the* **-i** *option could easily erase all the files in your current directory. The* **-i** *option will first prompt the user to confirm whether the file should be deleted.*

Matching Single Characters

The question mark (**?**) matches only a single character in filenames. Suppose you want to match the files **doc1** and **docA**, but not the file **document**. Whereas the asterisk will match filenames of any length, the question mark limits the match to just one extra character.

The next example matches files that begin with the word "doc" followed by a single differing letter:

```
$ ls
doc1 docA document
$ ls doc?
doc1 docA
```

Matching a Range of Characters

Whereas the * and ? file expansion characters specify incomplete portions of a filename, the brackets ([]) enable you to specify a set of valid characters to search for. Any character placed within the brackets will be matched in the filename. Suppose you want to list files beginning with "doc", but only ending in *1* or *A*. You are not interested in filenames ending in *2* or *B*, or any other character. Here is how it's done:

```
$ ls
doc1 doc2 doc3 docA docB docD document
$ ls doc[1A]
doc1 docA
```

You can also specify a set of characters as a range, rather than listing them one by one. A dash placed between the upper and lower bounds of a set of characters selects all characters within that range. The range is usually determined by the character set in use. In an ASCII character set, the range "a-g" will select all lowercase alphabetic characters from *a* through *g*. In the next example, files beginning with the pattern "doc" and ending in characters *1* through *3* are selected. Then, those ending in characters *B* through *E* are matched.

```
$ ls doc[1-3]
doc1 doc2 doc3
$ ls doc[B-E]
docB docD
```

You can combine the brackets with other file expansion characters to form flexible matching operators. Suppose you want to list only filenames ending in either a **.c** or **.o** extension, but no other extension. You can use a combination of the asterisk and brackets: ***. [co]**. The asterisk matches all filenames, and the brackets match only filenames with extension **.c** or **.o**.

```
$ ls *.[co]
main.c  main.o  calc.c
```

Matching Shell Symbols

At times, a file expansion character is actually part of a filename. In these cases, you need to quote the character by preceding it with a backslash to reference the file. In the next example, the user needs to reference a file that ends with the ? character, **answers?**. The ? is, however, a file expansion character and would match any filename beginning with "answers" that has one or more characters. In this case, the user quotes the ? with a preceding backslash to reference the filename.

```
$ ls answers\?
answers?
```

Placing the filename in double quotes will also quote the character.

```
$ ls "answers?"
answers?
```

This is also true for filenames or directories that have white space characters like the space character. In this case you can either use the backslash to quote the space character in the file or directory name, or place the entire name in double quotes.

```
$ ls My\ Documents
My Documents
$ ls "My Documents"
My Documents
```


Generating Patterns

Though not a file expansion operation, **{}** is often useful for generating names that you can use to create or modify files and directories. The braces operation only generates a list of names. It does not match on existing filenames. Patterns are placed within the braces and separated with commas. Any pattern placed within the braces will generate a version of the pattern, using either the preceding or following pattern, or both. Suppose you want to generate a list of names beginning with "doc", but only ending in the patterns "ument", "final", and "draft". Here is how it's done:

```
$ echo doc{ument,final,draft}
document docfinal docdraft
```

Since the names generated do not have to exist, you could use the **{}** operation in a command to create directories, as shown here:

```
$ mkdir {fall,winter,spring}report
$ ls
fallreport springreport winterreport
```

Standard Input/Output and Redirection

The data in input and output operations is organized like a file. Data input at the keyboard is placed in a data stream arranged as a continuous set of bytes. Data output from a command or program is also placed in a data stream and arranged as a continuous set of bytes. This input data stream is referred to in Linux as the *standard input,* and the output data stream is called the *standard output.* There is also a separate output data stream reserved solely for error messages, called the *standard error* (see the section "Redirecting and Piping the Standard Error: >&, 2>" later in this chapter).

Because the standard input and standard output have the same organization as that of a file, they can easily interact with files. Linux has a redirection capability that lets you easily move data in and out of files. You can redirect the standard output so that, instead of displaying the output on a screen, you can save it in a file. You can also redirect the standard input away from the keyboard to a file, so that input is read from a file instead of from your keyboard.

When a Linux command is executed that produces output, this output is placed in the standard output data stream. The default destination for the standard output data stream is a device—in this case, the screen. *Devices*, such as the keyboard and screen, are treated as files. They receive and send out streams of bytes with the same organization as that of a byte-stream file. The screen is a device that displays a continuous stream of bytes. By default, the standard output will send its data to the screen device, which will then display the data.

For example, the **ls** command generates a list of all filenames and outputs this list to the standard output. Next, this stream of bytes in the standard output is directed to the screen device. The list of filenames is then printed on the screen. The **cat** command also sends output to the standard output. The contents of a file are copied to the standard output, whose default destination is the screen. The contents of the file are then displayed on the screen.

Redirecting the Standard Output: > and >>

Suppose that instead of displaying a list of files on the screen, you would like to save this list in a file. In other words, you would like to direct the standard output to a file rather than the screen. To do this, you place the output redirection operator, the greater-than sign (>), followed by the name of a file on the command line after the Linux command. Table 3-6 lists the different ways you can use the redirection operators. In the next example, the output of the **ls** command is redirected from the screen device to a file:

```
$ ls -l *.c > programlist
```

The redirection operation creates the new destination file. If the file already exists, it will be overwritten with the data in the standard output. You can set the **noclobber** feature to prevent overwriting an existing file with the redirection operation. In this case, the redirection operation to an existing file will fail. You can overcome the **noclobber** feature by placing an exclamation point after the redirection operator. You can place the **noclobber** command in a shell configuration file to make it an automatic default operation (see Chapter 5). The next example sets the **noclobber** feature for the BASH shell and then forces the overwriting of the **oldletter** file if it already exists:

```
$ set -o noclobber
$ cat myletter >! oldletter
```

Although the redirection operator and the filename are placed after the command, the redirection operation is not executed after the command. In fact, it is executed before the command. The redirection operation creates the file and sets up the redirection before it receives any data from the standard output. If the file already exists, it will be destroyed and replaced by a file of the same name. In effect, the command generating the output is executed only after the redirected file has been created.

In the next example, the output of the **ls** command is redirected from the screen device to a file. First the **ls** command lists files, and in the next command, **ls** redirects its file list to the **listf** file. Then the **cat** command displays the list of files saved in **listf**. Notice the list of files in **listf** includes the **listf** filename. The list of filenames generated by the **ls** command

Command	Execution
ENTER	Execute a command line.
;	Separate commands on the same command line.
command **** *opts args*	Enter backslash before pressing ENTER to continue entering a command on the next line.
`***command***`	Execute a command.
$(*command***)**	Execute a command.
Special Characters for Filename Expansion	**Execution**
*****	Match on any set of characters.
?	Match on any single characters.
[]	Match on a class of possible characters.
****	Quote the following character. Used to quote special characters.
Redirection	**Execution**
command **>** *filename*	Redirect the standard output to a file or device, creating the file if it does not exist and overwriting the file if it does exist.
command **<** *filename*	Redirect the standard input from a file or device to a program.
command **>>** *filename*	Redirect the standard output to a file or device, appending the output to the end of the file.
command **>!** *filename*	In the C shell and the Korn shell, the exclamation point forces the overwriting of a file if it already exists. This overrides the **noclobber** option.
command **2>** *filename*	Redirect the standard error to a file or device in the Bourne shell.
command **2>>** *filename*	Redirect and append the standard error to a file or device in the Bourne shell.
command **2>&1**	Redirect the standard error to the standard output in the Bourne shell.
command **>&** *filename*	Redirect the standard error to a file or device in the C shell.
Pipes	**Execution**
command **|** *command*	Pipe the standard output of one command as input for another command.
command **|&** *command*	Pipe the standard error as input to another command in the C shell.

TABLE 3-6 The Shell Operations

includes the name of the file created by the redirection operation—in this case, **listf**. The **listf** file is first created by the redirection operation, and then the **ls** command lists it along with other files.

```
$ ls
mydata intro preface
$ ls > listf
$ cat listf
mydata intro listf preface
```

TIP *Errors occur when you try to use the same filename for both an input file for the command and the redirected destination file. In this case, because the redirection operation is executed first, the input file, because it exists, is destroyed and replaced by a file of the same name. When the command is executed, it finds an input file that is empty.*

You can also append the standard output to an existing file using the **>>** redirection operator. Instead of overwriting the file, the data in the standard output is added at the end of the file. In the next example, the **myletter** and **oldletter** files are appended to the **alletters** file. The **alletters** file will then contain the contents of both **myletter** and **oldletter**.

```
$ cat myletter >> alletters
$ cat oldletter >> alletters
```

The Standard Input

Many Linux commands can receive data from the standard input. The standard input itself receives data from a device or a file. The default device for the standard input is the keyboard. Characters typed on the keyboard are placed in the standard input, which is then directed to the Linux command. Just as with the standard output, you can also redirect the standard input, receiving input from a file rather than the keyboard. The operator for redirecting the standard input is the less-than sign (**<**). In the next example, the standard input is redirected to receive input from the **myletter** file, rather than the keyboard device (use CTRL-D to end the typed input). The contents of **myletter** are read into the standard input by the redirection operation. Then the **cat** command reads the standard input and displays the contents of **myletter**.

```
$ cat < myletter
hello Christopher
How are you today
$
```

You can combine the redirection operations for both standard input and standard output. In the next example, the **cat** command has no filename arguments. Without filename arguments, the **cat** command receives input from the standard input and sends output to the standard output. However, in the example the standard input has been redirected to receive its data from a file, while the standard output has been redirected to place its data in a file.

```
$ cat < myletter > newletter
```

Pipes: |

You may find yourself in situations in which you need to send data from one command to another. In other words, you may want to send the standard output of a command to another command, not to a destination file. Suppose you want to send a list of your filenames to the

printer to be printed. You need two commands to do this: the **ls** command to generate a list of filenames and the **lpr** command to send the list to the printer. In effect, you need to take the output of the **ls** command and use it as input for the **lpr** command. You can think of the data as flowing from one command to another. To form such a connection in Linux, you use what is called a *pipe*. The *pipe operator* (|, the vertical bar character) placed between two commands forms a connection between them. The standard output of one command becomes the standard input for the other. The pipe operation receives output from the command placed before the pipe and sends this data as input to the command placed after the pipe. As shown in the next example, you can connect the **ls** command and the **lpr** command with a pipe. The list of filenames output by the **ls** command is piped into the **lpr** command.

```
$ ls | lpr
```

You can combine the **pipe** operation with other shell features, such as file expansion characters, to perform specialized operations. The next example prints only files with a **.c** extension. The **ls** command is used with the asterisk and ".c" to generate a list of filenames with the **.c** extension. Then this list is piped to the **lpr** command.

```
$ ls *.c | lpr
```

In the preceding example, a list of filenames was used as input, but what is important to note is that pipes operate on the standard output of a command, whatever that might be. The contents of whole files or even several files can be piped from one command to another. In the next example, the **cat** command reads and outputs the contents of the **mydata** file, which are then piped to the **lpr** command:

```
$ cat mydata | lpr
```

Linux has many commands that generate modified output. For example, the **sort** command takes the contents of a file and generates a version with each line sorted in alphabetic order. The **sort** command works best with files that are lists of items. Commands such as **sort** that output a modified version of its input are referred to as *filters*. Filters are often used with pipes. In the next example, a sorted version of **mylist** is generated and piped into the **more** command for display on the screen. Note that the original file, **mylist**, has not been changed and is not itself sorted. Only the output of **sort** in the standard output is sorted.

```
$ sort mylist | more
```

The standard input piped into a command can be more carefully controlled with the standard input argument (-). When you use the dash as an argument for a command, it represents the standard input.

Redirecting the Standard Error:2>, >>

When you execute commands, an error could possibly occur. You may give the wrong number of arguments, or some kind of system error could take place. When an error occurs, the system issues an error message. Usually such error messages are displayed on the screen, along with the standard output. Linux distinguishes between standard output and error

messages, however. Error messages are placed in yet another standard byte stream, called the *standard error.* In the next example, the **cat** command is given as its argument the name of a file that does not exist, **myintro**. In this case, the **cat** command simply issues an error:

```
$ cat myintro
cat : myintro not found
$
```

Because error messages are in a separate data stream from the standard output, error messages still appear on the screen for you to see even if you have redirected the standard output to a file. In the next example, the standard output of the **cat** command is redirected to the file **mydata**. However, the standard error, containing the error messages, is still directed to the screen.

```
$ cat myintro > mydata
cat : myintro not found
$
```

You can redirect the standard error, as you can the standard output. This means you can save your error messages in a file for future reference. This is helpful if you need a record of the error messages. Like the standard output, the standard error has the screen device for its default destination. However, you can redirect the standard error to any file or device you choose using special redirection operators. In this case, the error messages will not be displayed on the screen.

Redirection of the standard error relies on a special feature of shell redirection. You can reference all the standard byte streams in redirection operations with numbers. The numbers 0, 1, and 2 reference the standard input, standard output, and standard error, respectively. By default, an output redirection, **>**, operates on the standard output, 1. You can modify the output redirection to operate on the standard error, however, by preceding the output redirection operator with the number 2. In the next example, the **cat** command again will generate an error. The error message is redirected to the standard byte stream represented by the number 2, the standard error.

```
$ cat nodata 2> myerrors
$ cat myerrors
cat : nodata not found
$
```

You can also append the standard error to a file by using the number 2 and the redirection append operator (**>>**). In the next example, the user appends the standard error to the **myerrors** file, which then functions as a log of errors:

```
$ cat nodata 2>> myerrors
```

Jobs: Background, Kills, and Interruptions

In Linux, you not only have control over a command's input and output, but also over its execution. You can run a job in the background while you execute other commands. You can also cancel commands before they have finished executing. You can even interrupt a command,

starting it again later from where you left off. Background operations are particularly useful for long jobs. Instead of waiting at the terminal until a command finishes execution, you can place it in the background. You can then continue executing other Linux commands. You can, for example, edit a file while other files are printing. The background commands, as well as commands to cancel and interrupt jobs, are listed in Table 3-7.

Running Jobs in the Background

You execute a command in the background by placing an ampersand (**&**) on the command line at the end of the command. When you place a job in the background, a user job number and a system process number are displayed. The user job number, placed in brackets, is the number by which the user references the job. The system process number is the number by which the system identifies the job. In the next example, the command to print the file **mydata** is placed in the background:

```
$ lpr mydata &
[1]  534
$
```


Background Jobs	Execution
%*jobnum*	References job by job number, use the **jobs** command to display job numbers.
%	References recent job.
%*string*	References job by an exact matching string.
%?*string***?**	References job that contains unique string.
%--	References job before recent job.
&	Execute a command in the background.
fg %*jobnum*	Bring a command in the background to the foreground or resume an interrupted program.
bg	Place a command in the foreground into the background.
CTRL-Z	Interrupt and stop the currently running program. The program remains stopped and waiting in the background for you to resume it.
notify %*jobnum*	Notifies you when a job ends.
kill %*jobnum* **kill** *processnum*	Cancel and end a job running in the background.
jobs	List all background jobs.
ps -a	List all currently running processes, including background jobs.
at *time date*	Execute commands at a specified time and date. The time can be entered with hours and minutes, and qualified as A.M. or P.M.

TABLE 3-7 Job Management Operations

You can place more than one command in the background. Each is classified as a job and given a name and a job number. The command **jobs** lists the jobs being run in the background. Each entry in the list consists of the job number in brackets, whether it is stopped or running, and the name of the job. The **+** sign indicates the job currently being processed, and the **-** sign indicates the next job to be executed. In the next example, two commands have been placed in the background. The **jobs** command then lists those jobs, showing which one is currently being executed.

```
$ lpr intro &
[1]  547
$ cat *.c > myprogs &
[2]  548
$ jobs
[1]  +  Running  lpr intro
[2]  -  Running  cat *.c > myprogs
$
```

Referencing Jobs

Normally jobs are referenced using the job number, preceded by a **%** symbol. You can obtain this number with the **jobs** command, which will list all background jobs, as shown in the preceding example. In addition you can also reference a job using an identifying string (see Table 3-7). The string must be either an exact match or a partial unique match. If there is no exact or unique match, you will receive an error message. Also, the **%** symbol itself without any job number references the recent background job. Followed by a `--` it references the second previous background job. The following example brings job 1 in the previous example to the foreground.

```
fg %lpr
```

Job Notification

After you execute any command in Linux, the system tells you what background jobs, if you have any running, have been completed so far. The system does not interrupt any operation, such as editing, to notify you about a completed job. If you want to be notified immediately when a certain job ends, no matter what you are doing on the system, you can use the **notify** command to instruct the system to tell you. The **notify** command takes a job number as its argument. When that job is finished, the system interrupts what you are doing to notify you the job has ended. The next example tells the system to notify the user when job 2 finishes:

```
$ notify %2
```

Bringing Jobs to the Foreground

You can bring a job out of the background with the foreground command, **fg**. If only one job is in the background, the **fg** command alone will bring it to the foreground. If more than one job is in the background, you must use the job's number with the command. You place the job number after the **fg** command, preceded by a percent sign. A **bg** command, usually used for interrupted jobs, places a job in the background. In the next example, the second job is

brought into the foreground. You may not immediately receive a prompt again because the second command is now in the foreground and executing. When the command is finished executing, the prompt appears and you can execute another command.

```
$ fg %2
cat *.c > myprogs
$
```

Canceling Jobs

If you want to cancel a job running in the background, you can force it to end with the **kill** command. The **kill** command takes as its argument either the user job number or the system process number. The user job number must be preceded by a percent sign (**%**). You can find out the job number from the **jobs** command. In the next example, the **jobs** command lists the background jobs; then job 2 is canceled:

```
$ jobs
[1]   +  Running  lpr intro
[2]   -  Running  cat *.c > myprogs
$ kill %2
```

Suspending and Stopping Jobs

You can suspend a job and stop it with the CTRL-Z key. This places the job to the side until it is restarted. The job is not ended; it merely remains suspended until you want to continue. When you're ready, you can continue with the job in either the foreground or the background using the **fg** or **bg** command. The **fg** command restarts a suspended job in the foreground. The **bg** command places the suspended job in the background.

At times, you may need to place a currently running job in the foreground into the background. However, you cannot move a currently running job directly into the background. You first need to suspend it with CTRL-Z and then place it in the background with the **bg** command. In the next example, the current command to list and redirect **.c** files is first suspended with CTRL-Z. Then that job is placed in the background.

```
$ cat *.c > myprogs
^Z
$ bg
```

NOTE *You can also use CTRL-Z to stop currently running jobs like Vi, suspending them in the background until you are ready to resume them. The Vi session remains a stopped job in the background until resumed with the bg command.*

Ending Processes: ps and kill

You can also cancel a job using the system process number, which you can obtain with the **ps** command. The **ps** command will display your processes, and you can use a process number to end any running process. The **ps** command displays a great deal more information than the **jobs** command does. The next example lists the processes a user is running. The PID is

the system process number, also known as the process ID. TTY is the terminal identifier. The time is how long the process has taken so far. COMMAND is the name of the process.

```
$ ps
PID     TTY       TIME      COMMAND
523     tty24     0:05      sh
567     tty24     0:01      lpr
570     tty24     0:00      ps
```

You can then reference the system process number in a **kill** command. Use the process number without any preceding percent sign. The next example kills process 567:

```
$ kill 567
```

Check the **ps** Man page for more detailed information about detecting and displaying process information. To just display a PID number, use the output options **-o pid=.** Combined with the **-C** command option you can display just the PID for a particular command. If there is more than one process for that command, such as multiple bash shells, then all the PIDs will be displayed.

```
$ ps -C lpr -o pid=
567
```

For unique commands, those you know have only one process running, you can safely combine the previous command with the **kill** command to end the process on one line. This avoids interactively having to display and enter the PID to kill the process. The technique can be useful for noninteractive operations like **cron** (see Chapter 27) and helpful for ending open-ended operations like video recording. In the following example, a command using just one process, `getatse`, is ended in a single kill operation. The **getatsc** is an hdtv recording command. Backquotes are used to first execute the `ps` command to obtain the PID (see "Values from Linux Commands in Chapter 4).

```
kill `ps -C getatsc -o pid=`
```

The C Shell: Command Line Editing and History

The C shell was originally developed for use with BSD Unix. With Linux, it is available as an alternative shell, along with the Korn and Bourne shells. The C shell incorporates all the core commands used in the Bourne shell but differs significantly in more complex features such as shell programming. The C shell was developed after the Bourne shell and was the first to introduce new features such as command line editing and the history utility. The Korn shell then later incorporated many of these same features. Then the bash shell, in turn, incorporated many of the features of all these shells. However, the respective implementations differ significantly. The C shell has limited command line editing that allows you to perform a few basic editing operations. C shell command line editing is not nearly as powerful as Korn shell command line editing. The history utility allows you to execute and edit previous commands. The history utility works in much the same way in the Korn, BASH, Z, and C shells. However, their command names differ radically, and the C shell has a very different set of history editing operations.

On most Linux distributions, an enhanced version of the C shell is used, called TCSH. Most of the commands are similar. You can access the C shell with the command **csh,** which is a link to the TCSH shell. The traditional prompt for the C shell is the % symbol. On some Linux distributions the prompt may remain the unchanged **$**.

```
$ csh
%
```

The command for entering the TCSH shell is **tcsh.**

C Shell Command Line Editing

Like the BASH shell, the C shell has only limited command line editing capabilities. They are, however, more powerful than those of the Bourne shell. Instead of deleting only a single character, you can delete a whole word. You can also perform limited editing operations using pattern substitution.

The CTRL-W key erases a recently entered word. The term "word" here is more of a technical concept that denotes how the shell parses a command. A word is parsed on a space or tab. Any character or set of characters surrounded by spaces or tabs is considered a word. With the CTRL-W key you can erase the text you have entered a word at a time.

```
% date who
% date
```

Other times you may need to change part of a word or several words in a command line. The C shell has a pattern substitution command that allows you to replace patterns in the command line. This substitution command is represented by a pattern enclosed in ^ symbols. The pattern to be replaced is enclosed between two ^. The replacement text immediately follows.

```
% ^pattern^newtext
```

The pattern substitution operation is not solely an editing command. It is also an execution command. Upon replacing the pattern, the corrected command will be displayed and then executed. In the next example, the date command has been misspelled. The shell displays an error message saying that such a command cannot be found. You can edit that command using the ^ symbols to replace the incorrect text. The command is then executed.

```
% dte
dte: not found
% ^dt^dat
date
Sun July 5 10:30:21 PST 1992
%
```

C Shell History Utility

As in the BASH shell, the C shell history utility keeps a record of the most recent commands you have executed. Table 3-8 lists the C shell history commands. The history utility keeps track of a limited number of the most recent commands, which are numbered from 1. The history utility

C Shell Event References	
!*event num*	References an event by its event number.
!*characters*	References an event beginning with specified characters.
!?*pattern*?	References an event containing the specified pattern.
!-*event num*	References an event with an offset from the first event.
!*num-num*	References a range of events.
C Shell Event Word References	
!*event num*:*word num*	References a particular word in an event.
!*event num*:^	References first argument (second word) in an event.
!*event num*:$	References last argument in an event.
!*event num*:^-$	References all arguments in an event.
!*event num*:*	References all arguments in an event.
C Shell Event Editing Substitutions	
!*event num*:s/*pattern*/*newtext*/	Edits an event with a pattern substitution. References a particular word in an event.
!*event num*:sg/*pattern*/*newtext*/	Performs a global substitution on all instances of a pattern in the event.
!*event num*:s/*pattern*/*newtext*/p	Suppresses execution of the edited event.

TABLE 3-8 C Shell History Commands

is not automatically turned on. You first have to define history with a **set** command and assign to it the number of commands you want recorded. This is often done as part of your shell configuration. In the next example, the history utility is defined and set to remember the last five commands.

```
% set history=5
```

As in the BASH shell, the commands remembered are referred to as events. To see the set of your most recent events, enter the word **history** on the command line and press ENTER. A list of your most recent commands is then displayed, with each event preceded by an event number.

```
% history
1 ls
2 vi mydata
3 mv mydata reports
4 cd reports
5 ls -F
```

Each of these events can be referenced by its event number, the beginning characters of the event, or a pattern of characters in the event. A pattern reference is enclosed in

question marks, **?.** You can re-execute any event using the history command **!.** The exclamation point is followed by an event reference such as an event number, beginning characters, or a pattern. In the next examples, the second command in the history list is referenced first by an event number, then by the beginning characters of the event, and then by a pattern in the event.

```
% !2
vi mydata

% !vi
vi mydata

% !?myd?
vi mydata
```


You can also reference a command using an offset from the end of the list. Preceding a number with a minus sign will offset from the end of the list to that command. In the next example, the second command, **vi mydata**, is referenced using an offset.

```
% !-4
vi mydata
```

An exclamation point is also used to identify the last command executed. It is equivalent to an offset of `-1`. In the next examples, both the offset of 1 and the exclamation point reference the last command, `Is -F`.

```
% !!
ls -F
mydata /reports

% !-1
ls -F
mydata /reports
```

C Shell History Event Substitutions

An event reference should be thought of as a representation of the characters making up the event. The event reference **!1** actually represents the characters "ls". As such, you can use an event reference as part of another command. The history operation can be thought of as a substitution. The characters making up the event replace the exclamation point and event reference entered on the command line. In the next example, the list of events is first displayed. Then a reference to the first event is used as part of a new command. The event reference `!1` evaluates to `ls`, becoming part of the command `ls > myfiles`.

```
% history
1 ls
2 vi mydata
3 mv mydata reports
4 cd reports
5 ls -F

% !1 > myfiles
ls > myfiles
```

You can also reference particular words in an event. An event is parsed into separated words, each word identified sequentially by a number starting from 0. An event reference followed by a colon and a number references a word in the event. The event reference **!3:2** references the second word in the third event. It first references the third event, **mv mydata reports**, and the second word in that event **mydata**. You can use such word references as part of a command. In the next example, **2:0** references the first word in the second event, **vi**, and replaces it with preface.

```
% !2:0 preface
vi preface
```

Using a range of numbers, you can reference several words in an event. The number of the first and last word in the range are separated by a dash. In the next example, **3:0-1** references the first two words of the third event, **mv mydata.**

```
% !3:0-1 oldletters
```

The metacharacters ^ and $ represent the second word and the last word in an event. They are used to reference arguments of the event. If you need just the first argument of an event, then ^ references it. **$** references the last argument. The range of ^-$ references all the arguments. (The first word, the command name, is not included.) In the next example, the arguments used in previous events are referenced and used as arguments in the current command. First, the first argument (the second word) in the second event, **mydata**, is used as an argument in an `lp` command, to print a file. Then, the last argument in the third event, **reports**, is used as an argument in the `ls` command, to list the filenames in reports. Then the arguments used in the third event, **mydata** and **reports**, are used as arguments in a **copy** command.

```
% lpr !2:^
lpr mydata

% ls !3:$
ls reports

% cp !3:^-$
cp mydata reports
```

The asterisk is a special symbol that represents all the arguments in a former command. It is equivalent to the range **^-$**. The last example can be rewritten using the asterisk, **!3***.

```
% cp !3*
cp mydata reports
```

In the C shell, whenever the exclamation point is used in a command, it is interpreted as a history command reference. If you need to use the exclamation point for other reasons, such as an electronic mail address symbol, you have to quote the exclamation point by placing a backslash in front of it.

```
% mail garnet\!chris < mydata
```

C Shell History Event Editing

You can edit history commands with a substitution command. The substitution command operates in the same way as the ^ command for command line editing. It replaces a pattern in a command with new text. To change a specific history command, enter an exclamation point and the event number of that command followed by a colon and the substitution command. The substitution command begins with the character **s** and is followed by a pattern enclosed in two slashes. The replacement text immediately follows, ending with a slash.

```
% !num:s/pattern/newtext/
```

In the next example, the pattern "my" in the third event is changed to "your". The changed event is then displayed and executed.

```
% history
1 ls
2 vi mydata
3 mv mydata reports
4 cd reports
5 ls -F
% !3:s/my/your/
mv yourdata reports
%
```

Preceding the **s** command with a **g** will perform a global substitution on an event. Every instance of the pattern in the event will be changed. In the next example, the extension of every filename in the first event is changed from **.c** to **.p** and then executed.

```
% lpr calc.c lib.c
% !1:gs/.c/.p/
lpr calc.p lib.p
%
```

The **&** command will repeat the previous substitution. In the next example the same substitution is performed on two commands, changing the filename **mydata** to **yourdata** in both the third and second events.

```
% !3:s/my/your/
mv yourdata reports
% !2:&
vi yourdata
```

When you perform a history operation on a command, it is automatically executed. You can suppress execution with a **p** qualifier. The **p** qualifier will only display the modified command, not execute it. This allows you to perform several operations on a command before you execute it. In the next example, two substitution commands are performed on the third command before it is executed.

```
% !3:s/mv/cp/:p          Does not execute the command
cp mydata reports
% !3:s/reports/books/    Changes and executes the command
cp mydata books
%
```

The TCSH Shell

The TCSH shell is essentially a version of the C shell with added features. It is fully compatible with the standard C shell and incorporates all of its capabilities, including the shell language and the history utility. TCSH has more advanced command line and history editing features than those found in the original C shell. You can use either Vi or Emacs key bindings to edit commands or history events. The TCSH shell also supports command line completion, automatically completing a command using just the few first characters you type in. TCSH shell has native language support, extensive terminal management, new built-in commands, and system variables. See the Man page for TCSH for more detailed information.

TCSH Command Line Completion

The command line has a built-in feature that performs command and filename completion. If you enter an incomplete pattern as a filename argument, you can press TAB to activate this feature, which will then complete the pattern to generate a filename. To use this feature, you type the partial name of the file on the command line and then press TAB. The shell will automatically look for the file with that partial prefix and complete it for you on the command line. In the next example, the user issues a **cat** command with an incomplete filename. When the user presses TAB, the system searches for a match and, upon finding one, fills in the filename.

```
> cat pre TAB
> cat preface
```

If more than one file has the same prefix, the shell will match the name as far as the filenames agree and then beep. You can then add more characters to select one or the other. For example:

```
> ls
document docudrama
> cat doc TAB
> cat docu beep
```

If, instead, you want a list of all the names that your incomplete filename matches, you can press CTRL-D on the command line. In the next example, the CTRL-D after the incomplete filename generates a list of possible filenames.

```
> cat doc Ctrl-d
document
docudrama
> cat docu
```

The shell redraws the command line, and you can then type in the remainder of the filename, or type in distinguishing characters, and press TAB to have the filename completed.

```
> cat docudrama
```

TCSH History Editing

As in the C shell, the TCSH shell's history utility keeps a record of the most recent commands you have executed. The history utility is a kind of short-term memory, keeping track of a limited number of the most recent commands. The history utility lets you reference a former

event by placing it on your command line and allowing you to execute it. However, you do not need to display the list first with history. The easiest way to do this is to use your UP ARROW and DOWN ARROW keys to place history events on your command line one at a time. Pressing the UP ARROW key once will place the last history event on your command line. Pressing it again places the next history event on your command line. The DOWN ARROW key will place the next command on the command line.

You can also edit the command line. The LEFT ARROW and RIGHT ARROW keys move you along the command line. You can then insert text wherever you stop your cursor. With the BACKSPACE and DELETE keys, you can delete characters. CTRL-A moves your cursor to the beginning of the command line, and CTRL-E moves it to the end. CTRL-K deletes the remainder of a line from the position of the cursor, and CTRL-U erases the entire line.

The Z-Shell

The Z-shell includes all of the features of the Korn shell and adds command line and history event features. The Z-shell performs automatic expansion on the command line after it has been parsed. Expansions are performed on filenames, processes, parameters, commands, arithmetic expressions, braces, and filename generation.

The Z-shell supports the use of Vi and Emacs key bindings for referencing history events, much like the BASH shell does. The UP ARROW and CTRL-P move you up to the previous event, and the DOWN ARROW and CTRL-N move you down to the next one. ESC < moves you to the first event and ESC > moves you to the last. The RIGHT and LEFT ARROWS move through an event line. CTRL-R CTRL-X performs a search of the history events.

History events can also be referenced using the **!** symbol, much like C shell history. When you enter the history command, a list of previous commands (called events) will be displayed, each with a number. To reference an event, enter the **!** symbol and its number. The following example references the third event.

```
!3
```

You can reference an event in several ways. You can use an offset from the current command, use a pattern to identify an event, or specify the beginning characters of an event. Table 3-9 lists these alternatives.

You can use word designators to include just segments of a history event in your command. A word designator indicates which word or words of a given command line will be included in a history reference. A colon separates the event number from the word designator. It can be omitted if the word designator begins with a **^**, **$**, ***** , **-**, or **%**. The words are numbered from 0, with 0 referring to the first word in an event, and 1 to the second word. **$** references the last word. A caret, **^**, references the first argument, the first word after the command word (same as 1). You can reference a range of words or, with *****, the remaining words in an event. To reference all the words from the third one to the end, use **3***. The ***** by itself references all the arguments (from 1 on). The following example references the second, third, and fourth words in the sixth event.

```
!6:2-4
```

Z-Shell History Commands	
!	Starts a history substitution, except when followed by a blank, newline, =, or (.
!!	Refers to the previous command. By itself, repeats the previous command.
!*num*	Refers to command line *num*.
!-*num*	Refers to the current command line minus num.
!*str*	Refers to the most recent command starting with str.
!?*str*[?]	Refers to the most recent command containing.
!#	Refers to the current command line typed so far.
!{...}	Insulates a history reference from adjacent characters (if necessary).
Z-Shell Word Designators	
0	The first input word (command).
num	The *num*th argument.
^	The first argument, that is, 1.
$	The last argument.
%	The word matched by (the most recent) ?*str* search.
str-str	A range of words; -str abbreviates 0-str.
*	All the arguments, or a null value if there is just one word in the event.
*str**	Abbreviates str-$.
str-	Like str* but omitting word $.

TABLE 3-9 Z-Shell History

4 CHAPTER

The Shell Scripts and Programming

A shell script combines Linux commands in such a way as to perform a specific task. The different kinds of shells provide many programming tools that you can use to create shell programs. You can define variables and assign values to them. You can also define variables in a script file and have a user interactively enter values for them when the script is executed. The shell provides loop and conditional control structures that repeat Linux commands or make decisions on which commands you want to execute. You can also construct expressions that perform arithmetic or comparison operations. All these shell programming tools operate in ways similar to those found in other programming languages, so if you're already familiar with programming, you might find shell programming simple to learn.

The BASH, TCSH, and Z shells described in Chapter 3 are types of shells. You can have many instances of a particular kind of shell. A *shell,* by definition, is an interpretive environment within which you execute commands. You can have many environments running at the same time, of either the same or different types of shells; you can have several shells running at the same time that are of the BASH shell type, for example.

This chapter will cover the basics of creating a shell program using the BASH and TCSH shells, the shells used on most Linux systems. You will learn how to create your own scripts, define shell variables, and develop user interfaces, as well as learn the more difficult task of combining control structures to create complex programs. Tables throughout the chapter list shell commands and operators, and numerous examples show how they are implemented.

Usually, the instructions making up a shell program are entered into a script file that can then be executed. You can even distribute your program among several script files, one of which will contain instructions on how to execute others. You can think of variables, expressions, and control structures as tools you use to bring together several Linux commands into one operation. In this sense, a shell program is a new and complex Linux command that you have created.

The BASH shell has a flexible and powerful set of programming commands that allows you to build complex scripts. It supports variables that can be either local to the given shell or exported to other shells. You can pass arguments from one script to another. The BASH shell has a complete set of control structures, including loops and `if` statements as well as

case structures, all of which you'll learn about as you read this book. All shell commands interact easily with redirection and piping operations that allow them to accept input from the standard input or send it to the standard output. Unlike the Bourne shell, the first shell used for Unix, BASH incorporates many of the features of the TCSH and Z shells. Arithmetic operations in particular are easier to perform in BASH.

The TCSH shell, like the BASH shell, also has programming language capabilities. You can define variables and assign values to them. You can place variable definitions and Linux commands in a script file and then execute that script. You can use loop and conditional control structures to repeat Linux commands or make decisions on which commands you want to execute. You can also place traps in your program to handle interrupts.

The TCSH shell differs from other shells in that its control structures conform more to a programming-language format. For example, the test condition for a TCSH shell's control structure is an expression that evaluates to true or false, not to a Linux command. A TCSH shell expression uses the same operators as those found in the C programming language. You can perform a variety of assignment, arithmetic, relational, and bitwise operations. The TCSH shell also allows you to declare numeric variables that can easily be used in such operations.

Shell Variables

Within each shell, you can enter and execute commands. You can further enhance the capabilities of a shell using shell variables. With a shell variable, you can hold data that you can reference over and over again as you execute different commands within a given shell. For example, you can define a shell variable to hold the name of complex filename. Then, instead of retyping the filename in different commands, you can reference it with the shell variable.

You define variables within a shell, and such variables are known as *shell variables.* Some utilities, such as the Mail utility, have their own shells with their own shell variables. You can also create your own shell using what are called *shell scripts.* You have a user shell that becomes active as soon as you log in. This is often referred to as the *login shell.* Special system-level parameter variables are defined within this login shell. Shell variables can also be used to define a shell's environment.

NOTE *Shell variables exist as long as your shell is active—that is, until you exit the shell. For example, logging out will exit the login shell. When you log in again, any variables you may need in your login shell must be defined again.*

Definition and Evaluation of Variables: =, $, set, unset

You define a variable in a shell when you first use the variable's name. A variable's name may be any set of alphabetic characters, including the underscore. The name may also include a number, but the number cannot be the first character in the name. A name may not have any other type of character, such as an exclamation point, an ampersand, or even a space. Such symbols are reserved by the shell for its own use. Also, a variable name may not include more than one word. The shell uses spaces on the command line to distinguish different components of a command such as options, arguments, and the name of the command.

You assign a value to a variable with the assignment operator (=). You type the variable name, the assignment operator, and then the value assigned. Do not place any spaces around the assignment operator. The assignment operation **poet = Virgil**, for example, will fail. (The C shell has a slightly different type of assignment operation.) You can assign any set of characters to a variable. In the next example, the variable **poet** is assigned the string **Virgil**:

```
$ poet=Virgil
```

Once you have assigned a value to a variable, you can then use the variable name to reference the value. Often you use the values of variables as arguments for a command. You can reference the value of a variable using the variable name preceded by the **$** operator. The dollar sign is a special operator that uses the variable name to reference a variable's value, in effect evaluating the variable. Evaluation retrieves a variable's value, usually a set of characters. This set of characters then replaces the variable name on the command line. Wherever a **$** is placed before the variable name, the variable name is replaced with the value of the variable. In the next example, the shell variable **poet** is evaluated and its contents, **Virgil**, are then used as the argument for an **echo** command. The **echo** command simply echoes or prints a set of characters to the screen.

```
$ echo $poet
Virgil
```

You must be careful to distinguish between the evaluation of a variable and its name alone. If you leave out the **$** operator before the variable name, all you have is the variable name itself. In the next example, the **$** operator is absent from the variable name. In this case, the **echo** command has as its argument the word "poet", and so prints out "poet":

```
$ echo poet
poet
```

The contents of a variable are often used as command arguments. A common command argument is a directory pathname. It can be tedious to retype a directory path that is being used over and over again. If you assign the directory pathname to a variable, you can simply use the evaluated variable in its place. The directory path you assign to the variable is retrieved when the variable is evaluated with the **$** operator. The next example assigns a directory pathname to a variable and then uses the evaluated variable in a copy command. The evaluation of **ldir** (which is **$ldir**) results in the pathname **/home/chris/letters**. The copy command evaluates to **cp myletter /home/chris/letters**.

```
$ ldir=/home/chris/letters
$ cp myletter $ldir
```

You can obtain a list of all the defined variables with the **set** command. If you decide you do not want a certain variable, you can remove it with the **unset** command. The **unset** command undefines a variable.

Variable Values: Strings

The values that you assign to variables may consist of any set of characters. These characters may be a character string that you explicitly type in or the result obtained from executing a Linux command. In most cases, you will need to quote your values using either single quotes,

double quotes, backslashes, or back quotes. Single quotes, double quotes, and backslashes allow you to quote strings in different ways. Back quotes have the special function of executing a Linux command and using its results as arguments on the command line.

Quoting Strings: Double Quotes, Single Quotes, and Backslashes

Variable values can be made up of any characters. However, problems occur when you want to include characters that are also used by the shell as operators. Your shell has certain metacharacters that it uses in evaluating the command line. A space is used to parse arguments on the command line. The asterisk, question mark, and brackets are metacharacters used to generate lists of filenames. The period represents the current directory. The dollar sign, **$**, is used to evaluate variables, and the greater-than (**>**) and less-than (**<**) characters, are redirection operators. The ampersand is used to execute background commands and the bar pipes output. If you want to use any of these characters as part of the value of a variable, you first need to quote them. Quoting a metacharacter on a command line makes it just another character. It is not evaluated by the shell.

You can use double quotes, single quotes, and backslashes to quote such metacharacters. Double and single quotes allow you to quote several metacharacters at a time. Any metacharacters within double or single quotes are quoted. A backslash quotes the single character that follows it.

If you want to assign more than one word to a variable, you need to quote the spaces separating the words. You can do so by enclosing all the words within double quotes. You can think of this as creating a character string to be assigned to the variable. Of course, any other metacharacters enclosed within the double quotes are also quoted.

In the following first example, the double quotes enclose words separated by spaces. Because the spaces are enclosed within double quotes, they are treated as characters, not as delimiters used to parse command line arguments. In the second example, double quotes also enclose a period, treating it as just a character. In the third example, an asterisk is also enclosed within the double quotes. The asterisk is considered just another character in the string and is not evaluated.

```
$ notice="The meeting will be tomorrow"
$ echo $notice
The meeting will be tomorrow

$ message="The project is on time."
$ echo $message
The project is on time.

$ notice="You can get a list of files with ls *.c"
$ echo $notice
You can get a list of files with ls *.c
```

Double quotes, however, do not quote the dollar sign, the operator that evaluates variables. A **$** operator next to a variable name enclosed within double quotes will still be evaluated, replacing the variable name with its value. The value of the variable will then become part of the string, not the variable name. There may be times when you want a variable within quotes to be evaluated. In the next example, the double quotes are used so that the winner's name will be included in the notice.

```
$ winner=dylan
$ notice="The person who won is $winner"
$ echo $notice
The person who won is dylan
```

On the other hand, there may be times when you do not want a variable within quotes to be evaluated. In that case you have to use the single quotes. Single quotes suppress any variable evaluation and treat the dollar sign as just another character. In the next example, single quotes prevent the evaluation of the winner variable.

```
$ winner=dylan
$ result='The name is in the $winner variable'
$ echo $result
The name is in the $winner variable
```


If, in this case, the double quotes were used instead, an unintended variable evaluation would take place. In the next example, the characters "$winner" are interpreted as a variable evaluation.

```
$ winner=dylan
$ result="The name is in the $winner variable"
$ echo $result
The name is in the dylan variable
```

You can always quote any metacharacter, including the **$** operator, by preceding it with a backslash. The use of the backslash is to quote ENTER keys (newlines). The backslash is useful when you want to both evaluate variables within a string and include the **$** character. In the next example, the backslash is placed before the **$** to treat it as a dollar sign character: **\$.** At the same time the variable **$winner** is evaluated because the double quotes that are used do not quote the **$** operator.

```
$ winner=dylan
$ result="$winner won \$100.00"
$ echo $result
dylan won $100.00
```

Quoting Commands: Single Quotes

There are, however, times when you may want to use single quotes around a Linux command. Single quotes allow you to assign the written command to a variable. If you do so, you can then use that variable name as another name for the Linux command. Entering the variable name preceded by the **$** operator on the command line will execute the command. In the next example, a shell variable is assigned the characters that make up a Linux command to list files, **'ls -F'**. Notice the single quotes around the command. When the shell variable is evaluated on the command line, the Linux command it contains will become a command line argument, and it will be executed by the shell.

```
$ lsf='ls -F'
$ $lsf
mydata /reports /letters
$
```

In effect you are creating another name for a command, like an alias.

Values from Linux Commands: Back Quotes

Although you can create variable values by typing in characters or character strings, you can also obtain values from other Linux commands. To assign the result of Linux command to a variable, you first need to execute the command. If you place a Linux command within back quotes on the command line, that command is first executed and its result becomes an argument on the command line. In the case of assignments, the result of a command can be assigned to a variable by placing the command within back quotes to first execute it. The back quotes can be thought of as an expression consisting of a command to be executed whose result is then assigned to the variable. The characters making up the command itself are not assigned. In the next example, the command **`ls *.c`** is executed and its result is then assigned to the variable **`listc`**. **`ls *.c`** generates a list of all files with a **`.c`** extension. This list of files is then assigned to the **`listc`** variable.

```
$ listc=`ls *.c`
$ echo $listc
main.c prog.c lib.c
```

You need to keep in mind the difference between single quotes and back quotes. Single quotes treat a Linux command as a set of characters. Back quotes force execution of the Linux command. There may be times when you accidentally enter single quotes when you mean to use back quotes. In the following first example, the assignment for the **`lscc`** variable has single quotes, not back quotes, placed around the **`ls *.c`** command. In this case, **`ls *.c`** are just characters to be assigned to the variable **`lscc`**. In the second example, back quotes are placed around the **`ls *.c`** command, forcing evaluation of the command. A list of filenames ending in **`.c`** is generated and assigned as the value of **`lscc`**.

```
$ lscc='ls *.c'
$ echo $lscc
ls *.c

$ lscc=`ls *.c`
$ echo $lscc
main.c  prog.c
```

Shell Scripts: User-Defined Commands

You can place shell commands within a file and then have the shell read and execute the commands in the file. In this sense, the file functions as a shell program, executing shell commands as if they were statements in a program. A file that contains shell commands is called a *shell script.*

You enter shell commands into a script file using a standard text editor such as the Vi editor. The **`sh`** or **`.`** command used with the script's filename will read the script file and execute the commands. In the next example, the text file called `lsc` contains an **`ls`** command that displays only files with the extension **`.c`**:

```
lsc
ls *.c
```

A run of the **lsc** script is shown here:

```
$ sh lsc
main.c calc.c
$ . lsc
main.c calc.c
```

Executing Scripts

You can dispense with the **sh** and **.** commands by setting the executable permission of a script file. When the script file is first created by your text editor, it is given only read and write permission. The **chmod** command with the **+x** option will give the script file executable permission. Once it is executable, entering the name of the script file at the shell prompt and pressing ENTER will execute the script file and the shell commands in it. In effect, the script's filename becomes a new shell command. In this way, you can use shell scripts to design and create your own Linux commands. You need to set the permission only once. In the next example, the **lsc** file's executable permission for the owner is set to on. Then the **lsc** shell script is directly executed like any Linux command.

```
$ chmod u+x lsc
$ lsc
main.c calc.c
```

You may have to specify that the script you are using is in your current working directory. You do this by prefixing the script name with a period and slash combination, **./**, as in **./lsc**. The period is a special character representing the name of your current working directory. The slash is a directory pathname separator. The following example shows how to execute the **lsc** script:

```
$ ./lsc
main.c calc.c
```

Script Arguments

Just as any Linux command can take arguments, so also can a shell script. Arguments on the command line are referenced sequentially starting with 1. An argument is referenced using the **$** operator and the number of its position. The first argument is referenced with **$1**, the second with **$2**, and so on. In the next example, the **lsext** script prints out files with a specified extension. The first argument is the extension. The script is then executed with the argument **c** (of course, the executable permission must have been set).

```
lsext
ls *.$1
```

A run of the **lsext** script with an argument is shown here:

```
$ lsext c
main.c calc.c
```

In the next example, the commands to print out a file with line numbers have been placed in an executable file called **lpnum**, which takes a filename as its argument. The **cat**

command with the **-n** option first outputs the contents of the file with line numbers. Then this output is piped into the **lpr** command, which prints it. The command to print out the line numbers is executed in the background.

```
lpnum
cat -n $1 | lpr &
```

A run of the **lpnum** script with an argument is shown here:

```
$ lpnum mydata
```

You may need to reference more than one argument at a time. The number of arguments used may vary. In **lpnum**, you may want to print out three files at one time and five files at some other time. The **$** operator with the asterisk, **$***, references all the arguments on the command line. Using **$*** enables you to create scripts that take a varying number of arguments. In the next example, **lpnum** is rewritten using **$*** so that it can take a different number of arguments each time you use it:

```
lpnum
cat -n $* | lpr &
```

A run of the **lpnum** script with multiple arguments is shown here:

```
$ lpnum mydata preface
```

TCSH Argument Array: argv

The TCSH/C shell uses a different set of argument variables to reference arguments. These are very similar to those used in the C programming language. When a TCSH shell script is invoked, all the words on the command line are parsed and placed in elements of an array called **argv.** The **argv[0]** array will hold the name of the shell script, and beginning with **argv[1]**, each element will hold an argument entered on the command line. In the case of shell scripts, **argv[0]** will always contain the name of the shell script. As with any array element, you can access the contents of an argument array element by preceding it with a **$** operator. For example, **$argv[1]** accesses the contents of the first element in the **argv** array, the first argument. In the **greetarg** script, a greeting is passed as the first argument on the command line. This first argument is accessed with **$argv[1]**.

```
greetarg
#
echo "The greeting you entered was: $argv[1]"
```

A run of the **greetarg** script follows:

```
% greetarg Hello
The greeting you entered was: Hello
```

Each word is parsed on the command line unless it's quoted. In the next example, the **greetarg** script is invoked with an unquoted string and then a quoted string. Notice that the quoted string, "Hello, how are you", is treated as one argument.

```
% greetarg Hello, how are you
The greeting you entered was: Hello,
% greetarg "Hello, how are you"
The greeting you entered was: Hello, how are you
```

If more than one argument is entered, the arguments can each be referenced with a corresponding element in the **argv** array. In the next example, the **myargs** script prints out four arguments. Four arguments are then entered on the command line.

```
myargs
#
echo "The first argument is: $argv[1]"
echo "The second argument is: $argv[2]"
echo "The third argument is: $argv[3]"
echo "The fourth argument is: $argv[4]"
```

The run of the **myargs** script is shown here:

```
% myargs Hello Hi yo "How are you"
The first argument is: Hello
The second argument is: Hi
The third argument is: yo
The fourth argument is: How are you
```


Environment Variables and Subshells: export and setenv

When you log in to your account, your Linux system generates your user shell. Within this shell, you can issue commands and declare variables. You can also create and execute shell scripts. However, when you execute a shell script, the system generates a subshell. You then have two shells, the one you logged in to and the one generated for the script. Within the script shell you can execute another shell script, which will then have its own shell. When a script has finished execution, its shell terminates and you enter back to the shell from which it was executed. In this sense, you can have many shells, each nested within the other.

Variables that you define within a shell are local to it. If you define a variable in a shell script, then, when the script is run, the variable is defined with that script's shell and is local to it. No other shell can reference it. In a sense, the variable is hidden within its shell.

To illustrate this situation more clearly, the next example will use two scripts, one of which is called from within the other. When the first script executes, it generates its own shell. From within this shell, another script is executed which, in turn, generates its own shell. In the next example, the user first executes the **dispfirst** script, which displays a first name. When the **dispfirst** script executes, it generates its own shell and then, within that shell, it defines the **firstname** variable. After it displays the contents of **firstname**, the script executes another script: **displast**. When **displast** executes, it generates its own shell. It defines the **lastname** variable within its shell and then displays the contents of **lastname**. It then tries to reference **firstname** and display its contents. It cannot do so because **firstname** is local to the dispfirst shell and cannot be referenced outside it. An error message is displayed indicating that for the **displast** shell, **firstname** is an undefined variable.

```
dispfirst
firstname="Charles"

echo "First name is $firstname"

displast
```

displast

```
lastname="Dickens"

echo "Last name is $lastname"
echo "$firstname $lastname"
```

The run of the **dispfirst** script is shown here:

```
$ dispfirst
First name is Charles
Last name is Dickens
 Dickens
sh: firstname: not found
$
```

dispfile

```
myfile="List"

echo "Displaying $myfile"
pr -t -n $myfile
```

printfile

```
printfile

myfile="List"

echo "Printing $myfile"
lp $myfile &
```

The run of the **dispfile** script is shown here:

```
$ dispfile
Displaying List
1 screen
2 modem
3 paper
Printing List
$
```

If you want the same value of a variable used both in a script's shell and a subshell, you can simply define the variable twice, once in each script, and assign it the same value. In the previous example, there is a **myfile** variable defined in **dispfile** and in **printfile**. The user executes the **b** script, which first displays the list file with line numbers. When the **dispfile** script executes, it generates its own shell and then, within that shell, it defines the **myfile** variable. After it displays the contents of the file, the script then executes another script, **printfile**. When **printfile** executes, it generates its own shell. It defines its own **myfile** variable within its shell and then sends a file to the printer.

What if you want to define a variable in one shell and have its value referenced in any subshell? For example, what if you want to define the **myfile** variable in the **dispfile** script and have its value, **List**, referenced from within the **printfile** script, rather than explicitly defining another variable in **printfile**? Since variables are local to the shell they are defined in,

there is no way you can do this with ordinary variables. However, there is a type of variable called an *environment variable* that allows its value to be referenced by any subshell. Environment variables constitute an environment for the shell and any subshell it generates, no matter how deeply nested.

You can define environment variables in the three major types of shells: Bourne, Korn, and C. However, the strategy used to implement environmental variables in the Bourne and Korn shells is very different from that of the C shell. In the Bourne and Korn shells, environmental variables are exported. That is to say, a copy of an environmental variable is made in each subshell. In a sense, if the **myfile** variable is exported, a copy is automatically defined in each subshell for you. In the C shell, on the other hand, an environmental variable is defined only once and can be directly referenced by any subshell.

Shell Environment Variables

In the Bourne, BASH, and Korn shells, an environment variable can be thought of as a regular variable with added capabilities. To make an environment variable, you apply the **export** command to a variable you have already defined. The **export** command instructs the system to define a copy of that variable for each new shell generated. Each new shell will have its own copy of the environment variable. This process is called *exporting variables*.

In the next example, the variable **myfile** is defined in the **dispfile** script. It is then turned into an environment variable using the **export** command. The **myfile** variable will consequently be exported to any subshell, such as that generated when **printfile** is executed.

```
dispfile
myfile="List"
export myfile

echo "Displaying $myfile"
pr -t -n $myfile

printfile

printfile
echo "Printing $myfile"
lp $myfile &
```

The run of the **dispfile** script is shown here:

```
$ dispfile
Displaying List
1 screen
2 modem
3 paper
Printing List
$
```

When **printfile** is executed it will be given its own copy of **myfile** and can reference that copy within its own shell. You no longer need to explicitly define another **myfile** variable in **printfile**.

It is a mistake to think of exported environment variables as global variables. A new shell can never reference a variable outside of itself. Instead, a copy of the variable with its value is generated for the new shell. You can think of exported variables as exporting their values to a shell, not themselves. For those familiar with programming structures, exported variables can be thought of as a form of call-by-value.

TCSH and C Shell Environment Variables

In the TCSH and C shells, an environment variable is defined using a separate definition command, **setenv**. In this respect, an environment variable is really a very different type of variable from that of a regular local variable. A C shell environment variable operates more like a global variable. It can be directly referenced by any subshell. This differs from the Bourne, BASH, and Korn shells in which only a copy of the environment variable is passed down and used by the subshell.

To define an environment variable you first enter the **setenv** command followed by the variable name and then the value. There is no assignment operator. In the next example, the **myfile** environment variable is defined and assigned the value **List**.

```
% setenv myfile list
```

```
dispfile
setenv myfile "List"

echo "Displaying $myfile"
cat -n $myfile

printfile
```

```
printfile
echo "Printing $myfile"
lpr $myfile &
```

The run of the **dispfile** script is shown here:

```
% dispfile
Displaying List
1 screen
2 modem
3 paper
Printing List
$
```

In the previous example, the variable **myfile** is defined as an environment variable in the **dispfile** script. Notice the use of the **setenv** command instead of **set**. The **myfile** variable can now be referenced in any subshell, such as that generated when **printfile** is executed.

When **printfile** is executed, it will be able to directly access the **myfile** variable defined in the shell of the **dispfile** script.

Control Structures

You can control the execution of Linux commands in a shell script with control structures. Control structures allow you to repeat commands and to select certain commands over others. A control structure consists of two major components: a test and commands. If the test is successful, then the commands are executed. In this way, you can use control structures to make decisions as to whether commands should be executed.

There are two different kinds of control structures: *loops* and *conditions*. A loop repeats commands, whereas a condition executes a command when certain conditions are met. The BASH shell has three loop control structures: **`while`**, **`for`**, and **`for-in`**. There are two condition structures: **`if`** and **`case`**. The control structures have as their test the execution of a Linux command. All Linux commands return an exit status after they have finished executing. If a command is successful, its exit status will be 0. If the command fails for any reason, its exit status will be a positive value referencing the type of failure that occurred. The control structures check to see if the exit status of a Linux command is 0 or some other value. In the case of the **`if`** and **`while`** structures, if the exit status is a 0 value, then the command was successful and the structure continues.

Test Operations

With the **`test`** command, you can compare integers, compare strings, and even perform logical operations. The command consists of the keyword **`test`** followed by the values being compared, separated by an option that specifies what kind of comparison is taking place. The option can be thought of as the operator, but it is written, like other options, with a minus sign and letter codes. For example, **`-eq`** is the option that represents the equality comparison. However, there are two string operations that actually use an operator instead of an option. When you compare two strings for equality, you use the equal sign (**`=`**). For inequality you use **`!=`**. Table 4-1 lists some of the commonly used options and operators used by **`test`**. The syntax for the **`test`** command is shown here:

```
test value -option value
test string = string
```

In the next example, the user compares two integer values to see if they are equal. In this case, you need to use the equality option, **`-eq`**. The exit status of the **`test`** command is examined to find out the result of the test operation. The shell special variable **`$?`** holds the exit status of the most recently executed Linux command.

```
$ num=5
$ test $num -eq 10
$ echo $?
1
```

Instead of using the keyword **`test`** for the **`test`** command, you can use enclosing brackets. The command **`test $greeting = "hi"`** can be written as

```
$ [ $greeting = "hi" ]
```

Similarly, the test command **`test $num -eq 10`** can be written as

```
$ [ $num -eq 10 ]
```

Integer Comparisons	Function
`-gt`	Greater-than
`-lt`	Less-than
`-ge`	Greater-than-or-equal-to
`-le`	Less-than-or-equal-to
`-eq`	Equal
`-ne`	Not-equal
String Comparisons	
`-z`	Tests for empty string
`=`	Tests for equality of strings
`!=`	Tests for inequality of strings
Logical Operators	
`-a`	Logical AND
`-o`	Logical OR
`!`	Logical NOT
File Tests	
`-f`	File exists and is a regular file
`-s`	File is not empty
`-r`	File is readable
`-w`	File can be written to and modified
`-x`	File is executable
`-d`	Filename is a directory name

TABLE 4-1 BASH Shell Test Operators

The brackets themselves must be surrounded by white space: a space, TAB, or ENTER. Without the spaces, they are invalid.

Conditional Control Structures

The BASH shell has a set of conditional control structures that allow you to choose what Linux commands to execute. Many of these are similar to conditional control structures found in programming languages, but there are some differences. The **`if`** condition tests the success of a Linux command, not an expression. Furthermore, the end of an **`if-then`** command must be indicated with the keyword **`fi`**, and the end of a **`case`** command is indicated with the keyword **`esac`**. The condition control structures are listed in Table 4-2.

The **`if`** structure places a condition on commands. That condition is the exit status of a specific Linux command. If a command is successful, returning an exit status of 0, then the commands within the **`if`** structure are executed. If the exit status is anything other than 0,

Condition Control Structures: `if`, `else`, `elif`, `case`	**Function**
`if` *command* **`then`** *command* **`fi`**	**`if`** executes an action if its test command is true.
`if` *command* **`then`** *command* **`else`** *command* **`fi`**	**`if-else`** executes an action if the exit status of its test command is true; if false, then the **`else`** action is executed.
`if` *command* **`then`** *command* **`elif`** *command* **`then`** *command* **`else`** *command* **`fi`**	**`elif`** allows you to nest **`if`** structures, enabling selection among several alternatives; at the first true **`if`** structure, its commands are executed and control leaves the entire **`elif`** structure.
`case` *string* **`in`** *pattern* `)` *command* `;;` **`esac`**	**`case`** matches the string value to any of several patterns; if a pattern is matched, its associated commands are executed.
command `&&` *command*	The logical AND condition returns a true 0 value if both commands return a true 0 value; if one returns a nonzero value, then the AND condition is false and also returns a nonzero value.
command `||` *command*	The logical OR condition returns a true 0 value if one or the other command returns a true 0 value; if both commands return a nonzero value, then the OR condition is false and also returns a nonzero value.
`!` *command*	The logical NOT condition inverts the return value of the command.
Loop Control Structures: `while`, `until`, `for`, `for-in`, `select`	
`while` *command* **`do`** *command* **`done`**	**`while`** executes an action as long as its test command is true.
`until` *command* **`do`** *command* **`done`**	**`until`** executes an action as long as its test command is false.

TABLE 4-2 BASH Shell Control Structures

Loop Control Structures: while, until, for, for-in, select	
for *variable* **in** *list-values* **do** *command* **done**	**for-in** is designed for use with lists of values; the variable operand is consecutively assigned the values in the list.
for *variable* **do** *command* **done**	**for** is designed for reference script arguments; the variable operand is consecutively assigned each argument value.
select *string* **in** *item-list* **do** *command* **done**	**select** creates a menu based on the items in the *item-list*; then it executes the command; the command is usually a **case**.

TABLE 4-2 BASH Shell Control Structures (*continued*)

then the command has failed and the commands within the **if** structure are not executed. The **if** command begins with the keyword **if** and is followed by a Linux command whose exit condition will be evaluated. The keyword **fi** ends the command. The **elsels** script in the next example executes the **ls** command to list files with two different possible options, either by size or with all file information. If the user enters an **s**, files are listed by size; otherwise, all file information is listed.

```
elsels
echo Enter s to list file by sizes
echo       otherwise all file information is listed.
echo -n "Please enter option: "
read choice
if [  "$choice" = s  ]
    then
        ls -s
    else
            ls -l
fi
echo Good-bye
```

A run of the program follows:

```
$ elsels
Enter s to list file sizes,
otherwise all file information is listed.
Please enter option: s
total 2
    1 monday     2 today
$
```

Loop Control Structures

The **while** loop repeats commands. A **while** loop begins with the keyword **while** and is followed by a Linux command. The keyword **do** follows on the next line. The end of the loop is specified by the keyword **done**. The Linux command used in **while** structures is often a test command indicated by enclosing brackets.

The **for-in** structure is designed to reference a list of values sequentially. It takes two operands: a variable and a list of values. The values in the list are assigned one by one to the variable in the **for-in** structure. Like the **while** command, the **for-in** structure is a loop. Each time through the loop, the next value in the list is assigned to the variable. When the end of the list is reached, the loop stops. Like the **while** loop, the body of a **for-in** loop begins with the keyword **do** and ends with the keyword **done**. The **cbackup** script makes a backup of each file and places it in a directory called **sourcebak**. Notice the use of the ***** special character to generate a list of all filenames with a **.c** extension.

```
cbackup
for backfile in *.c
do
    cp $backfile sourcebak/$backfile
 echo $backfile
done
```

A run of the program follows:

```
$ cbackup
io.c
lib.c
main.c
$
```

The **for** structure without a specified list of values takes as its list of values the command line arguments. The arguments specified on the command line when the shell file is invoked become a list of values referenced by the **for** command. The variable used in the **for** command is set automatically to each argument value in sequence. The first time through the loop, the variable is set to the value of the first argument. The second time, it is set to the value of the second argument.

TCSH/C Shell Control Structures

As in other shells, the TCSH shell has a set of control structures that let you control the execution of commands in a script. There are loop and conditional control structures with which you can repeat Linux commands or make decisions about which commands you want to execute. The **while** and **if** control structures are more general purpose control structures, performing iterations and making decisions using a variety of different tests. The **switch** and **foreach** control structures are more specialized operations. The **switch** structure is a restricted form of the **if** condition that checks to see if a value is equal to one of a set of possible values. The **foreach** structure is a limited type of loop that runs through a list of values, assigning a new value to a variable with each iteration.

The TCSH shell differs from other shells in that its control structures conform more to a programming-language format. The test condition for a TCSH shell control structure is an expression that evaluates to true or false, not a Linux command. One key difference between BASH shell and TCSH shell control structures is that TCSH shell structures cannot redirect or pipe their output. They are strictly control structures, controlling the execution of commands.

Test Expressions

The **if** and **while** control structures use an expression as their test. A true test is any expression that results in a nonzero value. A false test is any expression that results in a 0 value. In the TCSH shell, relational and equality expressions can be easily used as test expressions, because they result in 1 if true and 0 if false. There are many possible operators that you can use in an expression. You can use a number of operators in an expression, as shown in Table 4-3. The test expression can also be arithmetic or a string comparison, but strings can only be compared for equality or inequality.

Unlike the BASH shell, you must enclose the TCSH shell **if** and **while** test expressions within parentheses. The next example shows a simple test expression testing to see if two strings are equal.

```
if ( $greeting == "hi" ) then
    echo Informal Greeting
endif
```

The TCSH shell has a separate set of operators for testing strings against other strings or against regular expressions. The **==** and **!=** operators test for the equality and inequality of strings. The **=~** and **!~** operators test a string against a regular expression and test to see if a pattern match is successful. The regular expression can contain any of the shell special characters. In the next example, any value of **greeting** that begins with an upper or lowercase **h** will match the regular expression **[Hh]***.

```
if ( $greeting =~ [Hh]* ) then
    echo Informal Greeting
endif
```

Like the BASH shell, the TCSH shell has several special operators that test the status of files. Many of these operators are the same. In the next example, the **if** command tests to see if the file **mydata** is readable.

```
if ( -r mydata ) then
    echo Informal Greeting
endif
```

TCSH Shell Conditions: if-then, if-then-else, switch

The TCSH shell has a set of conditional control structures with which you make decisions about what Linux commands to execute. Many of these conditional control structures are similar to conditional control structures found in the BASH shell. There are, however, some key differences. The TCSH shell **if** structure ends with the keyword **endif**. The **switch** structure uses the keyword **case** differently. It ends with the keyword **endsw** and uses the

String Comparisons	Function/Description
`==`	Tests for equality of strings
`!=`	Tests for inequality of strings
`=~`	Compares string to a pattern to test if equal; the pattern can be any regular expression
`!~`	Compares string to a pattern to test if not equal; the pattern can be any regular expression
Logical Operators	
`&&`	Logical AND
`||`	Logical OR
`!`	Logical NOT
File Tests	
`-e`	File exists
`-r`	File is readable
`-w`	File can be written to, modified
`-x`	File is executable
`-d`	Filename is a directory name
`-f`	File is an ordinary file
`-o`	File is owned by user
`-z`	File is empty
Relational Operators	
`>`	Greater than
`<`	Less than
`>=`	Greater than or equal to
`<=`	Less than or equal to
`!=`	Not equal
`==`	Equal

TABLE 4-3 TCSH Test Expression Operators

keyword **`breaksw`** instead of two semicolons. Furthermore, there are two **`if`** control structures: a simple version that executes only one command and a more complex version that can execute several commands as well as alternative commands. The simple version of **`if`** consists of the keyword **`if`** followed by a test and a single Linux command. The complex version ends with the keyword **`endif`**. The TCSH shell's conditional control structures are listed in Table 4-4.

Control Structures	Description
if(*expression*) **then** *commands* **endif**	If the expression is true, the following commands are executed. You can specify more than one Linux command.
if(*expression*) **then** *command* **else** *command* **endif**	If the expression is true, the command after **then** is executed. If the expression is false, the command following **else** is executed.
switch(*string*) **case** *pattern*: *command* **breaksw** **default**: *command* **endsw**	Allows you to choose among several alternative commands.

TABLE 4-4 TCSH Conditional Control Structures

The if-then Structure

The **if-then** structure places a condition on several Linux commands. That condition is an expression. If the expression results in a value other than 0, the expression is true and the commands within the **if** structure are executed. If the expression results in a 0 value, the expression is false and the commands within the **if** structure are not executed.

The **if-then** structure begins with the keyword **if** and is followed by an expression enclosed in parentheses. The keyword then follows the expression. You can then specify any number of Linux commands on the following lines. The keyword **endif** ends the **if** command. Notice that, whereas in the BASH shell the **then** keyword is on a line of its own, in the TCSH shell, **then** is on the same line as the test expression. The syntax for the **if-then** structure is shown here:

```
if ( Expression ) then
        Commands
    endif
```

The **ifls** script shown next allows you to list files by size. If you enter an **s** at the prompt, each file in the current directory is listed, followed by the number of blocks it uses. If you enter anything else at the prompt, the **if** test fails and the script does nothing.

ifls

```
#
echo -n "Please enter option: "
set option = $<

if ($option == "s") then
        echo Listing files by size
        ls -s
    endif
```

A run of the **ifls** script is shown here:

```
% ifls
Please enter option: s
Listing files by size
total 2
    1 monday     2 today
```

Often, you need to choose between two alternatives based on whether an expression is true. The **else** keyword allows an **if** structure to choose between two alternative commands. If the expression is true, those commands immediately following the test expression are executed. If the expression is false, those commands following the **else** keyword are executed. The syntax for the **if-else** command is shown here:

```
if ( expression ) then
        commands
    else
        commands
endif
```

The **elsels** script in the next example executes the **ls** command to list files with two different possible options: by size or with all file information. If the user enters an **s**, files are listed by size; otherwise, all file information is listed.

elsels

```
#
echo Enter s to list file sizes.
echo otherwise all file information is listed.
echo -n "Please enter option : "
set option = $<

if ($option == "s") then
            ls -s
        else
            ls -l
endif
echo Goodbye
```

A run of the **elsels** script follows:

```
> elsels
Enter s to list file sizes,
otherwise all file information is listed.
Please enter option: s
total 2
    1 monday     2 today
Good-bye
```

The switch Structure

The **switch** structure chooses among several possible alternative commands. It is similar to the BASH shell's case structure in that the choice is made by comparing a string with several possible patterns. Each possible pattern is associated with a set of commands. If a match is found, the associated commands are performed.

The **switch** structure begins with the keyword **switch** followed by a test string within parentheses. The string is often derived from a variable evaluation. A set of patterns then follows—each pattern preceded by the keyword **case** and terminated with a colon. Commands associated with this choice are listed after the colon. The commands are terminated with the keyword **breaksw**. After all the listed patterns, the keyword **endsw** ends the switch structure. The syntax for the switch structure is shown here:

```
switch (test-string)
    case pattern:
            commands
            breaksw
    case pattern:
            commands
            breaksw
    default:
            commands
            breaksw
    endsw
```

TCSH Shell Loops: while, foreach, repeat

The TCSH shell has a set of loop control structures that allow you to repeat Linux commands: **while**, **foreach**, and **repeat**. The TCSH shell loop control structures are listed in Table 4-5.

The **while** structure operates in a way similar to corresponding structures found in programming languages. Like the TCSH shell's **if** structure, the **while** structure tests the result of an expression. The TCSH shell's **foreach** structure, like the **for** and **for-in** structures in the BASH shell, does not perform any tests. It simply progresses through a list of values, assigning each value in turn to a specified variable. In this respect, the **foreach** structure is very different from corresponding structures found in programming languages. The **repeat** structure is a simple and limited control structure. It repeats one command a specified number of times. It has no test expression, and it cannot repeat more than one command.

Loop Control Structures	Description
while(*expression*) *command* **end**	Executes commands as long as the expression is true.
foreach *variable (arg-list)* *command* **end**	Iterates the loop for as many arguments as exist in the argument list. Each time through the loop, the variable is set to the next argument in the list; operates like **for-in** in the BASH shell.
repeat *num command*	Repeats a command the specified number of times.
continue	Jumps to next iteration, skipping the remainder of the loop commands.
break	Breaks out of a loop.

TABLE 4-5 TCSH Loop Control Structures

The while Structure

The **while** loop repeats commands. A **while** loop begins with the keyword **while** and is followed by an expression enclosed in parentheses. The end of the loop is specified by the keyword **end**. The syntax for the while loop is shown here:

```
while ( expression  )
        commands
    end
```

The **while** structure can easily be combined with a **switch** structure to drive a menu.

The foreach Structure

The **foreach** structure is designed to sequentially reference a list of values. It is very similar to the BASH shell's **for-in** structure. The **foreach** structure takes two operands: a variable and a list of values enclosed in parentheses. Each value in the list is assigned to the variable in the **foreach** structure. Like the **while** structure, the **foreach** structure is a loop. Each time through the loop, the next value in the list is assigned to the variable. When the end of the list is reached, the loop stops. Like the **while** loop, the body of a **foreach** loop ends with the keyword **end**. The syntax for the **foreach** loop is shown here:

```
foreach variable ( list of values )
    commands
end
```

In the **mylist** script, in the next example, the script simply outputs a list of each item with today's date. The list of items makes up the list of values read by the **foreach** loop. Each item is consecutively assigned to the variable **grocery**.

```
mylist
#
set tdate=`date '+%D'`
foreach grocery ( milk cookies apples cheese )
     echo "$grocery    $tdate"
end

$ mylist
milk       12/23/96
cookies    12/23/96
apples     12/23/96
cheese     12/23/96
$
```

The **foreach** loop is useful for managing files. In the **foreach** structure, you can use shell special characters in a pattern to generate a list of filenames for use as your list of values. This generated list of filenames then becomes the list referenced by the **foreach** structure. An asterisk by itself generates a list of all files and directories. ***.c** lists files with the **.c** extension. These are usually C source code files. The next example makes a backup of each file and places the backup in a directory called **sourcebak**. The pattern ***.c** generates a list of filenames that the **foreach** structure can operate on.

```
cbackup
#
foreach backfile ( *.c )
    cp $backfile sourcebak/$backfile
    echo $backfile
end
```

```
% cbackup
io.c
lib.c
main.c
```

The **foreach** structure without a specified list of values takes as its list of values the command line arguments. The arguments specified on the command line when the shell file was invoked become a list of values referenced by the **foreach** structure. The variable used in the **foreach** structure is set automatically to each argument value in sequence. The first time through the loop, the variable is set to the value of the first argument. The second time, it is set to the value of the second argument, and so on.

In the **mylistarg** script in the next example, there is no list of values specified in the **foreach** loop. Instead, the **foreach** loop consecutively reads the values of command line arguments into the **grocery** variable. When all the arguments have been read, the loop ends.

```
mylistarg
#
set tdate=`date '+%D'`
foreach grocery ( $argv[*] )
     echo "$grocery     $tdate"
end
```

```
$ mylistarg milk cookies apples cheese
milk       12/23/96
cookies    12/23/96
apples     12/23/96
cheese     12/23/96
$
```

CHAPTER 5
Shell Configuration

Four different major shells are commonly used on Linux systems: the Bourne Again shell (BASH), the AT&T Korn shell, the TCSH shell, and the Z shell. The BASH shell is an advanced version of the Bourne shell, which includes most of the advanced features developed for the Korn shell and the C shell. TCSH is an enhanced version of the C shell, originally developed for BSD versions of Unix. The AT&T Unix Korn shell is open source. The Z shell is an enhanced version of the Korn shell. Although their Unix counterparts differ greatly, the Linux shells share many of the same features. In Unix, the Bourne shell lacks many capabilities found in the other Unix shells. In Linux, however, the BASH shell incorporates all the advanced features of the Korn shell and C shell, as well as the TCSH shell. All four shells are available for your use, though the BASH shell is the default.

The BASH shell is the default shell for most Linux distributions. If you are logging in to a command line interface, you will be placed in the default shell automatically and given a shell prompt at which to enter your commands. The shell prompt for the BASH shell is a dollar sign (**$**). In a GUI interface, such as GNOME or KDE, you can open a terminal window that will display a command line interface with the prompt for the default shell (BASH). Though you log in to your default shell or display it automatically in a terminal window, you can change to another shell by entering its name. **tcsh** invokes the TCSH shell, **bash** the BASH shell, **ksh** the Korn shell, and **zsh** the Z shell. You can leave a shell by pressing CTRL-D or using the **exit** command. You only need one type of shell to do your work. Table 5-1 shows the different commands you can use to invoke different shells. Some shells have added links you can use the invoke the same shell, like **sh** and **bsh**, which link to and invoke the **bash** command for the BASH shell.

This chapter describes common features of the BASH shell, such as aliases, as well as how to configure the shell to your own needs using shell variables and initialization files. The other shells share many of the same features and use similar variables and initialization files.

Though the basic shell features and configurations are shown here, you should consult the respective online manuals and FAQs for each shell for more detailed examples and explanations (see Table 3-1 in Chapter 3 for the websites for each shell).

Shells	Description
`bash`	BASH shell, **/bin/bash**
`bsh`	BASH shell, **/bin/bsh** (link to **/bin/bash**)
`sh`	BASH shell, **/bin/sh** (link to **/bin/bash**)
`tcsh`	TCSH shell, **/usr/tcsh**
`csh`	TCSH shell, **/bin/csh** (link to **/bin/tcsh**)
`ksh`	Korn shell, **/bin/ksh** (also added link **/usr/bin/ksh**)
`zsh`	Z shell, **/bin/zsh**

TABLE 5-1 Shell Invocation Command Names

Shell Initialization and Configuration Files

Each type of shell has its own set of initialization and configuration files. The BASH shell configuration files were discussed previously. The TCSH shell uses **.login**, **.tcshrc**, and **.logout** files in place of **.bash_profile**, **.bashrc**, and **.bash_logout**. Instead of **.bash_profile**, some distributions use the name **.profile**. The Z shell has several initialization files: **.zshenv**, **.zlogin**, **.zprofile**, **.zschrc**, and **.zlogout**. See Table 5-2 for a listing. Check the Man pages for each shell to see how they are usually configured. When you install a shell, default versions of these files are automatically placed in the users' home directories. Except for the TCSH shell, all shells use much the same syntax for variable definitions and assigning values (TCSH uses a slightly different syntax, described in its Man pages).

Configuration Directories and Files

Applications often install configuration files in a user's home directory that contain specific configuration information, which tailors the application to the needs of that particular user. This may take the form of a single configuration file that begins with a period, or a directory that contains several configuration files. The directory name will also begin with a period. For example, Mozilla installs a directory called **.mozilla** in the user's home directory that contains configuration files. On the other hand, many mail application uses a single file called **.mailrc** to hold alias and feature settings set up by the user, though others like Evolution also have their own, **.evolution**. Most single configuration files end in the letters **rc**. **FTP** uses a file called **.netrc**. Most newsreaders use a file called **.newsrc**. Entries in configuration files are usually set by the application, though you can usually make entries directly by editing the file. Applications have their own set of special variables to which you can define and assign values. You can list the configuration files in your home directory with the `ls -a` command.

Filename	Function
BASH Shell	
.bash_profile	Login initialization file
.profile	Login initialization file (same as **.bash_profile**)
.bashrc	BASH shell configuration file
.bash_logout	Logout name
.bash_history	History file
/etc/profile	System login initialization file
/etc/bashrc	System BASH shell configuration file
/etc/profile.d	Directory for specialized BASH shell configuration files
TCSH Shell	
.login	Login initialization file
.tcshrc	TCSH shell configuration file
.logout	Logout file
Z Shell	
.zshenv	Shell login file (first read)
.zprofile	Login initialization file
.zlogin	Shell login file
.zshrc	Z shell configuration file
.zlogout	Logout file
Korn Shell	
.profile	Login initialization file
.kshrc	Korn shell configuration file

TABLE 5-2 Shell Configuration Files

Aliases

You use the **alias** command to create another name for a command. The **alias** command operates like a macro that expands to the command it represents. The alias does not literally replace the name of the command; it simply gives another name to that command. An **alias** command begins with the keyword **alias** and the new name for the command, followed by an equal sign and the command the alias will reference.

NOTE *No spaces can be around the equal sign used in the* **alias** *command.*

In the next example, **list** becomes another name for the **ls** command:

```
$ alias list=ls
$ ls
mydata today
$ list
mydata today
$
```

Aliasing Commands and Options

You can also use an alias to substitute for a command and its option, but you need to enclose both the command and the option within single quotes. Any command you alias that contains spaces must be enclosed in single quotes as well. In the next example, the alias **lss** references the **ls** command with its **-s** option, and the alias **lsa** references the **ls** command with the **-F** option. The **ls** command with the **-s** option lists files and their sizes in blocks, and **ls** with the **-F** option places a slash after directory names. Notice how single quotes enclose the command and its option.

```
$ alias lss='ls -s'
$ lss
mydata 14   today  6   reports  1
$ alias lsa='ls -F'
$ lsa
mydata today reports/
$
```

Aliases are helpful for simplifying complex operations. In the next example, **listlong** becomes another name for the **ls** command with the **-l** option (the long format that lists all file information), as well as the **-h** option for using a human-readable format for file sizes. Be sure to encase the command and its arguments within single quotes so that they are taken as one argument and not parsed by the shell.

```
$ alias listlong='ls -lh'
$ listlong
-rw-r--r--    1 root   root   51K  Sep  18  2003 mydata
-rw-r--r--    1 root   root   16K  Sep  27  2003 today
```

Aliasing Commands and Arguments

You may often use an alias to include a command name with an argument. If you execute a command that has an argument with a complex combination of special characters on a regular basis, you may want to alias it. For example, suppose you often list just your source code and object code files—those files ending in either a **.c** or **.o**. You would need to use as an argument for **ls** a combination of special characters such as ***.[co]**. Instead, you can alias **ls** with the **.[co]** argument, giving it a simple name. In the next example, the user creates an alias called **lsc** for the command **ls.[co]**:

```
$ alias lsc='ls *.[co]'
$ lsc
main.c main.o lib.c lib.o
```

Aliasing Commands

You can also use the name of a command as an alias. This can be helpful in cases in which you should use a command only with a specific option. In the case of the **rm**, **cp**, and **mv** commands, the **-i** option should always be used to ensure an existing file is not overwritten. Instead of always being careful to use the **-i** option each time you use one of these commands, you can alias the command name to include the option. In the next example, the **rm**, **cp**, and **mv** commands have been aliased to include the **-i** option:

```
$ alias rm='rm -i'
$ alias mv='mv -i'
$ alias cp='cp -i'
```

The **alias** command by itself provides a list of all aliases that have been defined, showing the commands they represent. You can remove an alias by using the **unalias** command. In the next example, the user lists the current aliases and then removes the **lsa** alias:

```
$ alias
lsa=ls -F
list=ls
rm=rm -i
$ unalias lsa
```

Controlling Shell Operations

The BASH shell has several features that enable you to control the way different shell operations work. For example, setting the **noclobber** feature prevents redirection from overwriting files. You can turn these features on and off like a toggle, using the **set** command. The **set** command takes two arguments: an option specifying on or off and the name of the feature. To set a feature on, you use the **-o** option, and to set it off, you use the **+o** option. Here is the basic form:

```
$ set -o feature        turn the feature on
$ set +o feature        turn the feature off
```

Three of the most common features are **ignoreeof**, **noclobber**, and **noglob**. Table 5-3 lists these different features, as well as the **set** command. Setting **ignoreeof** enables a feature that prevents you from logging out of the user shell with CTRL-D. CTRL-D is not only

Features	Description
$ set -+o *feature*	BASH shell features are turned on and off with the **set** command; **-o** sets a feature on and **+o** turns it off: **$ set -o noclobber** *set noclobber on* **$ set +o noclobber** *set noclobber off*
ignoreeof	Disables CTRL-D logout
noclobber	Does not overwrite files through redirection
noglob	Disables special characters used for filename expansion: *****, **?**, **~**, and **[]**

TABLE 5-3 BASH Shell Special Features

used to log out of the user shell, but also to end user input entered directly into the standard input. CTRL-D is used often for the Mail program or for utilities such as **cat**. You can easily enter an extra CTRL-D in such circumstances and accidentally log yourself out. The **ignoreeof** feature prevents such accidental logouts. In the next example, the **ignoreeof** feature is turned on using the **set** command with the **-o** option. The user can then log out only by entering the **logout** command.

```
$ set -o ignoreeof
$ CTRL-D
Use exit to logout
$
```

Environment Variables and Subshells: export

When you log in to your account, Linux generates your user shell. Within this shell, you can issue commands and declare variables. You can also create and execute shell scripts. When you execute a shell script, however, the system generates a subshell. You then have two shells, the one you logged in to and the one generated for the script. Within the script shell, you can execute another shell script, which then has its own shell. When a script has finished execution, its shell terminates and you return to the shell from which it was executed. In this sense, you can have many shells, each nested within the other. Variables you define within a shell are local to it. If you define a variable in a shell script, then, when the script is run, the variable is defined with that script's shell and is local to it. No other shell can reference that variable. In a sense, the variable is hidden within its shell.

You can define environment variables in all types of shells, including the BASH shell, the Z shell, and the TCSH shell. The strategy used to implement environment variables in the BASH shell, however, is different from that of the TCSH shell. In the BASH shell, environment variables are exported. That is to say, a copy of an environment variable is made in each subshell. For example, if the **EDITOR** variable is exported, a copy is automatically defined in each subshell for you. In the TCSH shell, on the other hand, an environment variable is defined only once and can be directly referenced by any subshell.

In the BASH shell, an environment variable can be thought of as a regular variable with added capabilities. To make an environment variable, you apply the **export** command to a variable you have already defined. The **export** command instructs the system to define a copy of that variable for each new shell generated. Each new shell will have its own copy of the environment variable. This process is called *exporting variables.* To think of exported environment variables as global variables is a mistake. A new shell can never reference a variable outside of itself. Instead, a copy of the variable with its value is generated for the new shell.

Configuring Your Shell with Shell Parameters

When you log in, Linux will set certain parameters for your login shell. These parameters can take the form of variables or features. See the earlier section "Controlling Shell Operations" for a description of how to set features. Linux reserves a predefined set of variables for shell and system use. These are assigned system values, in effect setting parameters. Linux sets up parameter shell variables you can use to configure your user shell. Many of these parameter

shell variables are defined by the system when you log in. Some parameter shell variables are set by the shell automatically, and others are set by initialization scripts, described later. Certain shell variables are set directly by the shell, and others are simply used by it. Many of these other variables are application specific, used for such tasks as mail, history, or editing. Functionally, it may be better to think of these as system-level variables, as they are used to configure your entire system, setting values such as the location of executable commands on your system, or the number of history commands allowable. See Table 5-4 for a list of those shell variables set by the shell for shell-specific tasks; Table 5-5 lists those used by the shell for supporting other applications.

A reserved set of keywords is used for the names of these system variables. You should not use these keywords as the names of any of your own variable names. The system shell variables are all specified in uppercase letters, making them easy to identify. Shell feature variables are in lowercase letters. For example, the keyword **HOME** is used by the system to define the **HOME** variable. **HOME** is a special environment variable that holds the pathname of the user's home directory. On the other hand, the keyword **noclobber** is used to set the **noclobber** feature on or off.

Shell Parameter Variables

Many of the shell parameter variables automatically defined and assigned initial values by the system when you log in can be changed, if you wish. However, some parameter variables exist whose values should not be changed. For example, the **HOME** variable holds the

Shell Variables	Description
BASH	Holds full pathname of BASH command
BASH_VERSION	Displays the current BASH version number
GROUPS	Groups that the user belongs to
HISTCMD	Number of the current command in the history list
HOME	Pathname for user's home directory
HOSTNAME	The hostname
HOSTTYPE	Displays the type of machine the host runs on
OLDPWD	Previous working directory
OSTYPE	Operating system in use
PATH	List of pathnames for directories searched for executable commands
PPID	Process ID for shell's parent shell
PWD	User's working directory
RANDOM	Generates random number when referenced
SHLVL	Current shell level, number of shells invoked
UID	User ID of the current user

Table 5-4 Shell Variables Set by the Shell

Shell Variables	Description
BASH_VERSION	Displays the current BASH version number
CDPATH	Search path for the **cd** command
EXINIT	Initialization commands for Ex/Vi editor
FCEDIT	Editor used by the history **fc** command.
GROUPS	Groups that the user belongs to
HISTFILE	The pathname of the history file
HISTSIZE	Number of commands allowed for history
HISTFILESIZE	Size of the history file in lines
HISTCMD	Number of the current command in the history list
HOME	Pathname for user's home directory
HOSTFILE	Sets the name of the hosts file, if other than **/etc/hosts**
IFS	Interfield delimiter symbol
IGNOREEOF	If not set, EOF character will close the shell. Can be set to the number of EOF characters to ignore before accepting one to close the shell (default is 10)
INPUTRC	Set the **inputrc** configuration file for Readline (command line). Default is current directory, **.inputrc**. Most Linux distributions set this to **/etc/inputrc**
KDEDIR	The pathname location for the KDE desktop
LOGNAME	Login name
MAIL	Name of specific mail file checked by Mail utility for received messages, if MAILPATH is not set
MAILCHECK	Interval for checking for received mail
MAILPATH	List of mail files to be checked by Mail for received messages
HOSTTYPE	Linux platforms, such as i686, x86_64, or ppc
PROMPT_COMMAND	Command to be executed before each prompt, integrating the result as part of the prompt
HISTFILE	The pathname of the history file
PS1	Primary shell prompt
PS2	Secondary shell prompt
QTDIR	Location of the Qt library (used for KDE)
SHELL	Pathname of program for type of shell you are using
TERM	Terminal type
TMOUT	Time that the shell remains active awaiting input
USER	Username
UID	Real user ID (numeric)
EUID	Effective user ID (EUID, numeric). This is usually the same as the **UID** but can be different when the user changes IDs, as with the **su** command, which allows a user to become an effective root user

Table 5-5 System Environment Variables Used by the Shell

pathname for your home directory. Commands such as **cd** reference the pathname in the **HOME** shell variable to locate your home directory. Some of the more common of these parameter variables are described in this section. Other parameter variables are defined by the system and given an initial value that you are free to change. To do this, you redefine them and assign a new value. For example, the **PATH** variable is defined by the system and given an initial value; it contains the pathnames of directories where commands are located. Whenever you execute a command, the shell searches for it in these directories. You can add a new directory to be searched by redefining the **PATH** variable yourself, so that it will include the new directory's pathname. Still other parameter variables exist that the system does not define. These are usually optional features, such as the **EXINIT** variable that enables you to set options for the Vi editor. Each time you log in, you must define and assign a value to such variables. Some of the more common parameter variables are **SHELL**, **PATH**, **PS1**, **PS2**, and **MAIL**. The **SHELL** variable holds the pathname of the program for the type of shell you log in to. The **PATH** variable lists the different directories to be searched for a Linux command. The **PS1** and **PS2** variables hold the prompt symbols. The **MAIL** variable holds the pathname of your mailbox file. You can modify the values for any of them to customize your shell.

NOTE *You can obtain a listing of the currently defined shell variables using the* ***env*** *command. The* ***env*** *command operates like the* ***set*** *command, but it lists only parameter variables.*

Using Initialization Files

You can automatically define parameter variables using special shell scripts called initialization files. An *initialization file* is a specially named shell script executed whenever you enter a certain shell. You can edit the initialization file and place in it definitions and assignments for parameter variables. When you enter the shell, the initialization file will execute these definitions and assignments, effectively initializing parameter variables with your own values. For example, the BASH shell's **.bash_profile** file is an initialization file executed every time you log in. It contains definitions and assignments of parameter variables. However, the **.bash_profile** file is basically only a shell script, which you can edit with any text editor such as the Vi editor; changing, if you wish, the values assigned to parameter variables.

In the BASH shell, all the parameter variables are designed to be environment variables. When you define or redefine a parameter variable, you also need to export it to make it an environment variable. This means any change you make to a parameter variable must be accompanied by an **export** command. You will see that at the end of the login initialization file, **.bash_profile**, there is usually an **export** command for all the parameter variables defined in it.

Your Home Directory: HOME

The **HOME** variable contains the pathname of your home directory. Your home directory is determined by the parameter administrator when your account is created. The pathname for your home directory is automatically read into your **HOME** variable when you log in. In the next example, the **echo** command displays the contents of the **HOME** variable:

```
$ echo $HOME
/home/chris
```

The **HOME** variable is often used when you need to specify the absolute pathname of your home directory. In the next example, the absolute pathname of **reports** is specified using **HOME** for the home directory's path:

```
$ ls $HOME/reports
```

Command Locations: PATH

The **PATH** variable contains a series of directory paths separated by colons. Each time a command is executed, the paths listed in the **PATH** variable are searched one by one for that command. For example, the **cp** command resides on the system in the directory **/bin**. This directory path is one of the directories listed in the **PATH** variable. Each time you execute the **cp** command, this path is searched and the **cp** command located. The system defines and assigns **PATH** an initial set of pathnames. In Linux, the initial pathnames are **/bin** and **/usr/bin**.

The shell can execute any executable file, including programs and scripts you have created. For this reason, the **PATH** variable can also reference your working directory; so if you want to execute one of your own scripts or programs in your working directory, the shell can locate it. No spaces are allowed between the pathnames in the string. A colon with no pathname specified references your working directory. Usually, a single colon is placed at the end of the pathnames as an empty entry specifying your working directory. For example, the pathname **//bin:/usr/bin:** references three directories: **/bin**, **/usr/bin**, and your current working directory.

```
$ echo $PATH
/bin:/usr/sbin:
```

You can add any new directory path you want to the **PATH** variable. This can be useful if you have created several of your own Linux commands using shell scripts. You can place these new shell script commands in a directory you create and then add that directory to the **PATH** list. Then, no matter what directory you are in, you can execute one of your shell scripts. The **PATH** variable will contain the directory for that script, so that directory will be searched each time you issue a command.

You add a directory to the **PATH** variable with a variable assignment. You can execute this assignment directly in your shell. In the next example, the user **chris** adds a new directory, called **mybin,** to the **PATH**. Although you could carefully type in the complete pathnames listed in **PATH** for the assignment, you can also use an evaluation of **PATH**—**$PATH**—in their place. In this example, an evaluation of **HOME** is also used to designate the user's **home** directory in the new directory's pathname. Notice the empty entry between two colons, which specifies the working directory:

```
$ PATH=$PATH:$HOME/mybin:
$ export PATH
$ echo $PATH
/bin:/usr/bin::/home/chris/mybin
```

If you add a directory to **PATH** yourself while you are logged in, the directory will be added only for the duration of your login session. When you log back in, the login initialization file, **.bash_profile**, will again initialize your **PATH** with its original set of directories.

The **.bash_profile** file is described in detail a bit later in this chapter. To add a new directory to your **PATH** permanently, you need to edit your **.bash_profile** file and find the assignment for the **PATH** variable. Then, you simply insert the directory, preceded by a colon, into the set of pathnames assigned to **PATH**.

Specifying the BASH Environment: BASH_ENV

The **BASH_ENV** variable holds the name of the BASH shell initialization file to be executed whenever a BASH shell is generated. For example, when a BASH shell script is executed, the **BASH_ENV** variable is checked and the name of the script that it holds is executed before the shell script. The **BASH_ENV** variable usually holds **$HOME/.bashrc**. This is the **.bashrc** file in the user's home directory. (The **.bashrc** file is discussed later in this chapter.) You can specify a different file if you wish, using that instead of the **.bashrc** file for BASH shell scripts.

Configuring the Shell Prompt

The **PS1** and **PS2** variables contain the primary and secondary prompt symbols, respectively. The primary prompt symbol for the BASH shell is a dollar sign (**$**). You can change the prompt symbol by assigning a new set of characters to the **PS1** variable. In the next example, the shell prompt is changed to the **->** symbol:

```
$ PS1= '->'
-> export PS1
->
```

You can change the prompt to be any set of characters, including a string, as shown in the next example:

```
$ PS1="Please enter a command: "
Please enter a command: export PS1
Please enter a command: ls
mydata /reports
Please enter a command:
```

The **PS2** variable holds the secondary prompt symbol, which is used for commands that take several lines to complete. The default secondary prompt is **>**. The added command lines begin with the secondary prompt instead of the primary prompt. You can change the secondary prompt just as easily as the primary prompt, as shown here:

```
$ PS2="@"
```

Like the TCSH shell, the BASH shell provides you with a predefined set of codes you can use to configure your prompt. With them you can make the time, your username, or your directory pathname a part of your prompt. You can even have your prompt display the history event number of the current command you are about to enter. Each code is preceded by a \ symbol: **\w** represents the current working directory, **\t** the time, and **\u** your username; **\!** will display the next history event number. In the next example, the user adds the current working directory to the prompt:

```
$ PS1="\w $"
/home/dylan $
```

The codes must be included within a quoted string. If no quotes exist, the code characters are not evaluated and are themselves used as the prompt. **PS1=\w** sets the prompt to the characters **\w**, not the working directory. The next example incorporates both the time and the history event number with a new prompt:

```
$ PS1="\t \! ->"
```

The following table lists the codes for configuring your prompt:

Prompt Codes	Description
\!	Current history number
\$	Use **$** as prompt for all users except the root user, which has the **#** as its prompt
\d	Current date
\#	History command number for just the current shell
\h	Hostname
\s	Shell type currently active
\t	Time of day in hours, minutes, and seconds
\u	Username
\v	Shell version
\w	Full pathname of the current working directory
\W	Name of the current working directory
****	Displays a backslash character
\n	Inserts a newline
\[\]	Allows entry of terminal-specific display characters for features like color or bold font
nnn	Character specified in octal format

The default BASH prompt is **\s-\v\$** to display the type of shell, the shell version, and the **$** symbol as the prompt. Some distributions like Fedora and Red Hat have changed this to a more complex command consisting of the user, the hostname, and the name of the current working directory. The actual operation is carried out in the **/etc/bashrc** file discussed in the later section "The System **/etc/ bashrc** BASH Script and the **/etc/profile.d** Directory." A sample configuration is shown here. The **/etc/ bashrc** file uses USER, HOSTNAME, and PWD environment variables to set these values. A simple equivalent is show here with an @ sign in the hostname and a $ for the final prompt symbol. The home directory is represented with a tilde (**~**).

```
$ PS1="\u@\h:\w$"
richard@turtle.com:~$
```

Specifying Your News Server

Several shell parameter variables are used to set values used by network applications, such as web browsers or newsreaders. **NNTPSERVER** is used to set the value of a remote news

server accessible on your network. If you are using an ISP, the ISP usually provides a Usenet news server you can access with your newsreader applications. However, you first have to provide your newsreaders with the Internet address of the news server. This is the role of the **NNTPSERVER** variable. News servers on the Internet usually use the NNTP protocol. **NNTPSERVER** should hold the address of such a news server. For many ISPs, the news server address is a domain name that begins with **nntp**. The following example assigns the news server address **nntp.myservice.com** to the **NNTPSERVER** shell variable. Newsreader applications automatically obtain the news server address from **NNTPSERVER**. Usually, this assignment is placed in the shell initialization file, **.bash_profile**, so that it is automatically set each time a user logs in.

```
NNTPSERVER=news.myservice.com
export NNTPSERVER
```


Configuring Your Login Shell: .bash_profile

The **.bash_profile** file is the BASH shell's login initialization file, which can also be named **.profile** (as in SUSE or Ubuntu Linux). It is a script file that is automatically executed whenever a user logs in. The file contains shell commands that define system environment variables used to manage your shell. They may be either redefinitions of system-defined variables or definitions of user-defined variables. For example, when you log in, your user shell needs to know what directories hold Linux commands. It will reference the **PATH** variable to find the pathnames for these directories. However, first, the **PATH** variable must be assigned those pathnames. In the **.bash_profile** file, an assignment operation does just this. Because it is in the **.bash_profile** file, the assignment is executed automatically when the user logs in.

Exporting Variables

Parameter variables also need to be exported, using the **export** command, to make them accessible to any subshell you may enter. You can export several variables in one **export** command by listing them as arguments. Usually, the **.bash_profile** file ends with an **export** command with a list of all the variables defined in the file. If a variable is missing from this list, you may be unable to access it. Notice the **export** command at the end of the **.profile** file in the first example in the next section. You can also combine the assignment and **export** command into one operation as shown here for **NNTPSERVER**:

```
export NNTPSERVER=news.myservice.com
```

Variable Assignments

A copy of the standard **.bash_profile** file provided for you when your account is created is listed in the next example. Notice how **PATH** is assigned, as is the value of **$HOME**. Both **PATH** and **HOME** are parameter variables the system has already defined. **PATH** holds the pathnames of directories searched for any command you enter, and **HOME** holds the pathname of your home directory. The assignment **PATH=$PATH:$HOME/bin** has the effect of redefining **PATH** to include your **bin** directory within your home directory so that your **bin** directory will also be searched for any commands, including ones you create yourself, such as scripts or programs. Notice **PATH** is then exported, so that it can be accessed by any subshell.

```
.bash_profile
# .bash_profile

# Get the aliases and functions
if [  -f  ~/.bashrc ]; then
          .  ~/.bashrc
fi

# User specific environment and startup programs
PATH=$PATH:$HOME/bin
export  PATH
```

The root user version of **.bash_profile** adds an entry to unset the **USERNAME** variable, which contains the user's text name.

```
unset USERNAME
```

Should you want to have your home directory searched also, you can use any text editor to modify this line in your **.bash_profile** file to **PATH=$PATH:$HOME/bin:$HOME**, adding **:$HOME** at the end. In fact, you can change this entry to add as many directories as you want searched. If you add a colon at the end, then your current working directory will also be searched for commands. Making commands automatically executable in your current working directory could be a security risk, allowing files in any directory to be executed, instead of in certain specified directories. An example of how to modify your **.bash_profile** file is shown in the following section.

```
PATH=$PATH:$HOME/bin:$HOME:
```

Editing Your BASH Profile Script

Your **.bash_profile** initialization file is a text file that can be edited by a text editor, like any other text file. You can easily add new directories to your **PATH** by editing **.bash_profile** and using editing commands to insert a new directory pathname in the list of directory pathnames assigned to the **PATH** variable. You can even add new variable definitions. If you do so, however, be sure to include the new variable's name in the **export** command's argument list. For example, if your **.bash_profile** file does not have any definition of the **EXINIT** variable, you can edit the file and add a new line that assigns a value to **EXINIT**. The definition **EXINIT='set nu ai'** will configure the Vi editor with line numbering and indentation. You then need to add **EXINIT** to the **export** command's argument list. When the **.bash_profile** file executes again, the **EXINIT** variable will be set to the command **set nu ai**. When the Vi editor is invoked, the command in the **EXINIT** variable will be executed, setting the line number and auto-indent options automatically.

In the following example, the user's **.bash_profile** has been modified to include definitions of **EXINIT** and redefinitions of **PATH**, **PS1**, and **HISTSIZE**. The **PATH** variable has **$HOME**: added to its value. **$HOME** is a variable that evaluates to the user's home directory, and the ending colon specifies the current working directory, enabling you to execute commands that may be located in either the home directory or the working directory. The redefinition of **HISTSIZE** reduces the number of history events saved, from 1000 defined in the system's **.profile** file, to 30. The redefinition of the **PS1** parameter variable changes the prompt to include the pathname of the current working directory. Any changes you make to

parameter variables within your **.bash_profile** file override those made earlier by the system's **.profile** file. All these parameter variables are then exported with the **export** command.

.bash_profile

```
# .bash_profile
# Get the aliases and functions
if [ -f ~/.bashrc ];
 then
    . ~/.bashrc
fi
# User-specific environment and startup programs
PATH=$PATH:$HOME/bin:$HOME:
 unset USERNAME
HISTSIZE=30
NNTPSERVER=news.myserver.com
EXINIT='set nu ai'
PS1="\w \$"
export PATH HISTSIZE EXINIT PS1 NNTPSERVER
```

Manually Re-executing the .bash_profile Script

Although **.bash_profile** is executed each time you log in, it is not automatically re-execute after you make changes to it. The **.bash_profile** file is an initialization file that is executed *only* whenever you log in. If you want to take advantage of any changes you make to it without having to log out and log in again, you can re-execute **.bash_profile** with the dot (**.**) command. The **.bash_profile** file is a shell script and, like any shell script, can be executed with the **.** command.

```
$ . .bash_profile
```

Alternatively, you can use the **source** command to execute the **.bash_profile** initialization file or any initialization file such as **.login** used in the TCSH shell or **.bashrc**.

```
$ source .bash_profile
```

System Shell Profile Script

Your Linux system also has its own profile file that it executes whenever any user logs in. This system initialization file is simply called **profile** and is found in the **/etc** directory, **/etc/profile**. This file contains parameter variable definitions the system needs to provide for each user. A copy of the system's **profile** file follows at the end of this section. On some distributions, this will be a very simple file, and on others much more complex. Some distributions like Fedora and Red Hat use a **pathmunge** function to generate a directory list for the **PATH** variable. Normal user paths will lack the system directories (those with **sbin** in the path) but include the name of their home directory, along with **/usr/kerberos/bin** for Kerberos tools. The path generated for the root user (EUID of 0) will include both system and user application directories, adding **/usr/kerberos/sbin**, **/sbin**, **/usr/sbin**, and **/usr/local/sbin**, as well as the root user local application directory, **/root/bin**.

```
# echo $PATH
/usr/kerberos/bin/usr/local/bin:usr/sbin:/bin:/usr/X11R6/bin:/home/richard/bin
```

A special work-around is included for the Korn Shell to set the User and Effective User IDs (**EUID** and **UID**).

The **USER**, **MAIL**, and **LOGNAME** variables are then set, provided that **/usr/bin/id**, which provides the user ID, is executable. The **id** command with the **-un** option provides the user ID's text name only, like **chris** or **richard**.

HISTSIZE is also redefined to include a larger number of history events. An entry has been added here for the **NNTPSERVER** variable. Normally, a news server address is a value that needs to be set for all users. Such assignments should be made in the system's **/etc/profile** file by the system administrator, rather than in each individual user's own **.bash_profile** file.

NOTE *The **/etc/profile** file also executes any scripts in the directory **/etc/profile.d.** This design allows for a more modular structure. Rather than make entries by editing the **/etc/profile** file, you can just add a script to **profile.d** directory.*

The **/etc/profile** file also runs the **/etc/inputrc** file, which configures your command line editor. Here you will find key assignments for different tasks, such as moving to the end of a line or deleting characters. Global options are set as well. Keys are represented in hexadecimal format.

The number of aliases and variable settings needed for different applications would make the **/etc/profile** file much too large to manage. Instead, application- and task-specific aliases and variables are placed in separate configuration files located in the **/etc/profile.d** directory. There are corresponding scripts for both the BASH and C shells. The BASH shell scripts are run by **/etc/profile**. The scripts are named for the kinds of tasks and applications they configure. For example, on Red Hat, sets the file type color coding when the **ls** command displays files and directories. The vim.sh file sets the an alias for the **vi** command, executing **vim** whenever the user enters just **vi**. The kde.sh file sets the global environment variable **KDEDIR**, specifying the KDE applications directory, in this case **/usr**. The krb5.sh file adds the pathnames for Kerberos, **/usr/kerberos**, to the **PATH** variable. Files run by the BASH shell end in the extension **.sh**, and those run by the C shell have the extension **.csh**.

/etc/profile

```
# /etc/profile

# Systemwide environment and startup programs, for login setup
# Functions and aliases go in /etc/bashrc

pathmunge () {
          if ! echo $PATH | /bin/egrep -q "(^|:)$1($|:)" ; then
              if [ "$2" = "after" ] ; then
                    PATH=$PATH:$1
               else
                    PATH=$1:$PATH
              fi
        fi
}

# ksh workaround
if [ -z "$EUID" -a -x /usr/bin/id ]; then
```

```
          EUID=`id -u`
          UID=`id -ru`
fi

# Path manipulation
if [ "$EUID" = "0" ]; then
          pathmunge /sbin
          pathmunge /usr/sbin
          pathmunge /usr/local/sbin
fi

# No core files by default
ulimit -S -c 0 > /dev/null 2>&1

if [ -x /usr/bin/id ]; then
          USER="`id -un`"
          LOGNAME=$USER
          MAIL="/var/spool/mail/$USER"
fi

HOSTNAME=`/bin/hostname`
HISTSIZE=1000

if [ -z "$INPUTRC" -a ! -f "$HOME/.inputrc" ]; then
    INPUTRC=/etc/inputrc
fi

export PATH USER LOGNAME MAIL HOSTNAME HISTSIZE INPUTRC

for i in /etc/profile.d/*.sh ; do
    if [ -r "$i" ]; then
              . $i
    fi
done

unset i
unset pathmunge
```

Configuring the BASH Shell: .bashrc

The **.bashrc** file is a configuration file executed each time you enter the BASH shell or generate a subshell. If the BASH shell is your login shell, **.bashrc** is executed along with your **.bash_login** file when you log in. If you enter the BASH shell from another shell, the **.bashrc** file is automatically executed, and the variable and alias definitions it contains will be defined. If you enter a different type of shell, the configuration file for that shell will be executed instead. For example, if you were to enter the TCSH shell with the **tcsh** command, the **.tcshrc** configuration file would be executed instead of **.bashrc**.

The User .bashrc BASH Script

The **.bashrc** shell configuration file is actually executed each time you generate a BASH shell, such as when you run a shell script. In other words, each time a subshell is created, the **.bashrc** file is executed. This has the effect of exporting any local variables or aliases you

have defined in the **.bashrc** shell initialization file. The **.bashrc** file usually contains the definition of aliases and any feature variables used to turn on shell features. Aliases and feature variables are locally defined within the shell. But the **.bashrc** file defines them in every shell. For this reason, the **.bashrc** file usually holds aliases and options you want defined for each shell. In this example, the standard **.bashrc** for users include only the execution of the system **/etc/bashrc** file. As an example of how you can add your own aliases and options, aliases for the **rm**, **cp**, and **mv** commands and the shell **noclobber** and **ignoreeof** options have been added. For the root user **.bashrc**, the **rm**, **cp**, and **mv** aliases have already been included in the root's **.bashrc** file.

```
.bashrc
# Source global definitions
if [ -f /etc/bashrc ];
 then
    . /etc/bashrc
fi
set  -o ignoreeof
set  -o noclobber
alias rm='rm -i'
alias mv='mv -i'
alias cp='cp -i'
```

You can add any commands or definitions of your own to your **.bashrc** file. If you have made changes to **.bashrc** and you want them to take effect during your current login session, you need to re-execute the file with either the **.** or the **source** command.

```
$ . .bashrc
```

The System /etc/bashrc BASH Script and the /etc/profile.d Directory

Linux systems usually contain a system **bashrc** file executed for all users. The file contains certain global aliases and features needed by all users whenever they enter a BASH shell. This is located in the **/etc** directory, **/etc/bashrc**. A user's own **.bashrc** file, located in the home directory, contains commands to execute this system **.bashrc** file. The **./etc/bashrc** command in the previous example of **.bashrc** does just that. Currently the **/etc/bashrc** file sets the default shell prompt, one for a terminal window and another for a screen interface. Several other specialized aliases and variables are then set using configuration files located in the **/etc/profile.d** directory. These scripts are executed by **/etc/bashrc** if the shell is not the user login shell.

The BASH Shell Logout File: .bash_logout

The **.bash_logout** file is also a configuration file, but it is executed when the user logs out. It is designed to perform any operations you want done whenever you log out. Instead of variable definitions, the **.bash_logout** file usually contains shell commands that form a kind of shutdown procedure—actions you always want taken before you log out. One common logout command is to clear the screen and then issue a farewell message.

As with **.bash_profile**, you can add your own shell commands to **.bash_logout**. In fact, the **.bash_logout** file is not automatically set up for you when your account is first created.

You need to create it yourself, using the Vi or Emacs editor. You could then add a farewell message or other operations. In the next example, the user has a **clear** command and an **echo** command in the **.bash_logout** file. When the user logs out, the **clear** command clears the screen, and then the **echo** command displays the message "Good-bye for now."

.bash_logout

```
# ~/.bash_logout
clear
echo "Good-bye for now"
```


The TCSH Shell Configuration

The TCSH shell is essentially a version of the C shell with added features. Configuration operations perform much the same tasks but with slightly different syntax. The **alias** command operates the same but uses a different command format. System variables are assigned values using TCSH shell assignment operators, and the initialization and configuration files have different names.

TCSH/C Aliases

You use the **alias** command to create another name for a command. The alias operates like a macro that expands to the command it represents. The alias does not literally replace the name of the command; it simply gives another name to that command.

An **alias** command begins with the keyword alias and the new name for the command, followed by the command that the alias will reference. In the next example, the **ls** command is aliased with the name **list**. **list** becomes another name for the **ls** command.

```
> alias list ls
> ls
mydata intro
> list
mydata intro
>
```

Should the command you are aliasing have options, you will need to enclose the command and the option within single quotes. An aliased command that has spaces will need quotation marks as well. In the next example, **ls** with the **-l** option is given the alias **longl:**

```
> alias longl 'ls -l'
> ls -l
-rw-r--r--  1  chris weather 207 Feb  20  11:55   mydata
> longl
-rw-r--r--  1  chris weather 207 Feb  20  11:55   mydata
>
```

You can also use the name of a command as an alias. In the case of the **rm**, **cp**, and **mv** commands, the **-i** option should always be used to ensure that an existing file is not overwritten. Instead of always being careful to use the **-i** option each time you use one of

these commands, you can alias the command name to include the option. In the next examples, the **rm**, **cp**, and **mv** commands have been aliased to include the **-i** option.

```
> alias rm 'rm -i'
> alias mv 'mv -i'
> alias cp 'cm -i'
```

The **alias** command by itself provides a list of all aliases in effect and their commands. An alias can be removed with the **unalias** command.

```
> alias
lss   ls -s
list  ls
rm    rm -i
> unalias lss
```

TCSH/C Shell Feature Variables: Shell Features

The TCSH shell has several features that allow you to control how different shell operations work. The TCSH shell's features include those in the PDSKH shell as well as many of its own. For example, the TCSH shell has a **noclobber** option to prevent redirection from overwriting files. Some of the more commonly used features are **echo**, **noclobber**, **ignoreeof**, and **noglob**. The TCSH shell features are turned on and off by defining and undefining a variable associated with that feature. A variable is named for each feature, for example, the **noclobber** feature is turned on by defining the **noclobber** variable. You use the **set** command to define a variable and the **unset** command to undefine a variable. To turn on the **noclobber** feature you issue the command **set noclobber**. To turn it off you use the command **unset noclobber**.

```
> set feature-variable
> unset feature-variable
```

These variables are also sometimes referred to as toggles since they are used to turn features on and off.

echo

Setting **echo** enables a feature that displays a command before it is executed. The command **set echo** turns the **echo** feature on, and the command **unset echo** turns it off.

ignoreeof

Setting **ignoreeof** enables a feature that prevents users from logging out of the user shell with a CTRL-D. It is designed to prevent accidental logouts. With this feature turned off, you can log out by pressing CTRL-D. However, CTRL-D is also used to end user input entered directly into the standard input. It is used often for the Mail program or for utilities such as **cat**. You can easily enter an extra CTRL-D in such circumstances and accidentally log yourself out. The **ignoreeof** feature prevents such accidental logouts. When it is set, you have to explicitly log out, using the **logout** command:

```
$ set ignoreeof
$ ^D
Use logout to logout
$
```

noclobber

Setting **noclobber** enables a feature that safeguards existing files from redirected output. With the **noclobber** feature, if you redirect output to a file that already exists, the file will not be overwritten with the standard output. The original file will be preserved. There may be situations in which you use a name that you have already given to an existing file as the name for the file to hold the redirected output. The **noclobber** feature prevents you from accidentally overwriting your original file:

```
> set noclobber
> cat preface > myfile
myfile: file exists
$
```

There may be times when you want to overwrite a file with redirected output. In this case, you can place an exclamation point after the redirection operator. This will override the **noclobber** feature, replacing the contents of the file with the standard output:

```
> cat preface >! myfile
```

noglob

Setting **noglob** enables a feature that disables special characters in the user shell. The characters *****, **?**, **[]**, and **~** will no longer expand to matched filenames. This feature is helpful if, for some reason, you have special characters as part of a filename. In the next example, the user needs to reference a file that ends with the **?** character, **answers?**. First the user turns off special characters, using the **noglob** option. Now the question mark on the command line is taken as part of the filename, not as a special character, and the user can reference the **answers?** file.

```
$ set noglob
$ ls answers?
answers?
```

TCSH/C Special Shell Variables for Configuring Your System

As in the BASH shell, you can use special shell variables in the TCSH shell to configure your system. Some are defined initially by your system, and you can later redefine them with a new value. There are others that you must initially define yourself. One of the more commonly used special variables is the **prompt** variable that allows you to create your own command line prompts. Another is the **history** variable with which you determine how many history events you want to keep track of.

In the TCSH shell, many special variables have names and functions similar to those in the BASH or Public Domain Korn Shell (PDKSH) shells. Some are in uppercase, but most are written in lowercase. The **EXINIT** and **TERM** variables retain their uppercase form. However, **history** and **cdpath** are written in lowercase. Other special variables may perform similar functions but have very different implementations. For example, the **mail** variable holds the same information as the BASH **MAIL**, **MAILPATH**, and **MAILCHECK** variables together.

prompt, prompt2, prompt3

The **prompt**, **prompt2**, and **prompt3** variables hold the prompts for your command line. You can configure your prompt to be any symbol or string that you want. To have your

command line display a different symbol as a prompt, you simply use the **set** command to assign that symbol to the prompt variable. In the next example, the user assigns a + sign to the **prompt** variable, making it the new prompt.

```
> set prompt = "+"
+
```

You can use a predefined set of codes to make configuring your prompt easier. With them, you can make the time, your username, or your directory pathname a part of your prompt. You can even have your prompt display the history event number of the current command you are about to enter. Each code is preceded by a **%** symbol, for example, **%/** represents the current working directory, **%t** the time, and **%n** your username. **%!** will display the next history event number. In the next example, the user adds the current working directory to the prompt.

```
> set prompt = "%/ >"
/home/dylan >
```

The next example incorporates both the time and the history event number with a new prompt.

```
> set prompt = "%t %! $"
```

Here is a list of the codes:

%/	Current working directory
%h, %!, !	Current history number
%t	Time of day
%n	Username
%d	Day of the week
%w	Current month
%y	Current year

The prompt2 variable is used in special cases when a command may take several lines to input. **prompt2** is displayed for the added lines needed for entering the command. **prompt3** is the prompt used if the spell check feature is activated.

cdpath

The **cdpath** variable holds the pathnames of directories to be searched for specified subdirectories referenced with the **cd** command. These pathnames form an array just like the array of pathnames assigned to the TCSH shell **path** variable. Notice the space between the pathnames.

```
> set cdpath=(/usr/chris/reports /usr/chris/letters)
```

history and savehist

As you learned earlier, the **history** variable can be used to determine the number of history events you want saved. You simply assign to it the maximum number of events that **history** will record. When the maximum is reached, the count starts over again from 1. The **savehist** variable, however, holds the number of events that will be saved in the file **.history** when you log out. When you log in again, these events will become the initial history list.

In the next example, up to 20 events will be recorded in your history list while you are logged in. However, only the last 5 will be saved in the **.history** file when you log out. Upon logging in again, your history list will consist of your last 5 commands from the previous session.

```
> set history=20
> set savehist=5
```

mail

In the TCSH shell, the **mail** variable combines the features of the **MAIL**, **MAILCHECK**, and **MAILPATH** variables in the BASH and PDKSH shells. The TCSH shell **mail** variable is assigned as its value an array whose elements contain both the time interval for checking for mail and the directory pathnames for mailbox files to be checked. To assign values to these elements, you assign an array of values to the **mail** variable. The array of new values is specified with a list of words separated by spaces and enclosed in parentheses. The first value is a number that sets the number of seconds to wait before checking for mail again. This value is comparable to that held by the BASH shell's **MAILCHECK** variable. The remaining values consist of the directory pathnames of mailbox files that are to be checked for your mail. Notice that these values combine the functions of the BASH and Korn shells' **MAIL** and **MAILPATH** variables.

In the next example, the **mail** variable is set to check for mail every 20 minutes (1200 seconds), and the mailbox file checked is in **usr/mail/chris**. The first value in the array assigned to mail is 1200, and the second value in the array is the pathname of the mailbox file to be checked.

```
> set mail ( 1200 /usr/mail/chris )
```

You can, just as easily, add more mailbox file pathnames to the **mail** array. In the next example, two mailboxes are designated. Notice the spaces surrounding each element.

```
> set mail ( 1200 /usr/mail/chris /home/mail/chris )
```

TCSH/C Shell Initialization Files: .login, .tcshrc, .logout

The TCSH shell has three initialization files: **.login**, **.logout**, and **.tcshrc**. The files are named for the operation they execute. The **.login** file is a login initialization file that executes each time you log in. The **.logout** file executes each time you log out. The **.tcshrc** file is a shell initialization file that executes each time you enter the TCSH shell, either from logging in or by explicitly changing to the TCSH shell from another shell with the **tcsh** command.

.login

The TCSH shell has its own login initialization file called the **.login** file that contains shell commands and special variable definitions used to configure your shell. The **.login** file corresponds to the **.profile** file used in the BASH and PDKSH shells.

A **.login** file contains **setenv** commands that assign values to special environment variables, such as **TERM**. You can change these assigned values by editing the **.login** file with any of the standard editors. You can also add new values. Remember, however, that in the TCSH shell, the command for assigning a value to an environment variable is **setenv**. In the next example, the **EXINIT** variable is defined and assigned the Vi editor's line numbering and auto-indent options.

```
> setenv EXINIT 'set nu ai'
```

Be careful when editing your **.login** file. Inadvertent editing changes could cause variables to be set incorrectly or not at all. It is wise to make a backup of your **.login** file before editing it.

If you have made changes to your **.login** file and you want the changes to take effect during your current login session, you will need to re-execute the file. You do so using the **source** command. The **source** command will actually execute any initialization file, including the **.tcshrc** and **.logout** files. In the next example, the user re-executes the **.login** file.

```
> source .login
```

If you are also planning to use the PDKSH shell on your Linux system, you need to define a variable called **ENV** within your **.login** file and assign it the name of the PDKSH shell initialization file. If you should later decide to enter the PDKSH shell from your TCSH shell, the PDKSH shell initialization file can be located and executed for you. In the example of the **.login** file shown next, you will see that the last command sets the PDKSH shell initialization file to **.kshrc** to the **ENV** variable: **setenv ENV $HOME/.kshrc**.

```
.login
setenv term vt100
setenv EXINIT 'set nu ai'

setenv ENV $HOME/.kshrc
```

.tcshrc

The **.tcshrc** initialization file is executed each time you enter the TCSH shell or generate any subshell. If the TCSH shell is your login shell, then the **.tcshrc** file is executed along with your **.login** file when you log in. If you enter the TCSH shell from another shell, the **.tcshrc** file is automatically executed, and the variable and alias definitions it contains will be defined.

The **.tcshrc** shell initialization file is actually executed each time you generate a shell, such as when you run a shell script. In other words, each time a subshell is created, the **.tcshrc** file is executed. This allows you to define local variables in the **.tcshrc** initialization file and have them, in a sense, exported to any subshell. Even though such user-defined special variables as **history** are local, they will be defined for each subshell generated. In this way, **history** is set for each subshell. However, each subshell has its own local

history variable. You could even change the local **history** variable in one subshell without affecting any of those in other subshells. Defining special variables in the shell initialization file allows you to treat them like BASH shell exported variables. An exported variable in a BASH or PDKSH shell only passes a copy of itself to any subshell. Changing the copy does not affect the original definition.

The **.tcshrc** file also contains the definition of aliases and any feature variables used to turn on shell features. Aliases and feature variables are locally defined within the shell. But the **.tcshrc** file will define them in every shell. For this reason, **.tcshrc** usually holds such aliases as those defined for the **rm**, **cp**, and **mv** commands. The next example is a **.tcshrc** file with many of the standard definitions.


```
.tcshrc
set shell=/usr/bin/csh
set path= $PATH (/bin /usr/bin . )
set cdpath=( /home/chris/reports /home/chris/letters )

set prompt="! $cwd >"
set history=20

set ignoreeof
set noclobber

alias rm 'rm -i'
alias mv 'mv -i'
alias cp 'cm -i'
```

Local variables, unlike environment variables, are defined with the **set** command. Any local variables that you define in **.tcshrc** should use the **set** command. Any variables defined with **setenv** as environment variables, such as **TERM**, should be placed in the **.login** file. The next example shows the kinds of definitions found in the **.tcshrc** file. Notice that the **history** and **noclobber** variables are defined using the **set** command.

```
set history=20
set noclobber
```

You can edit any of the values assigned to these variables. However, when editing the pathnames assigned to **path** or **cdpath**, bear in mind that these pathnames are contained in an array. Each element in an array is separated by a space. If you add a new pathname, you need to be sure that there is a space separating it from the other pathnames.

If you have made changes to **.tcshrc** and you want them to take effect during your current login session, remember to re-execute the **.tcshrc** file with the **source** command:

```
> source .tcshrc
```

.logout

The **.logout** file is also an initialization file, but it is executed when the user logs out. It is designed to perform any operations you want done whenever you log out. Instead of variable definitions, the **.logout** file usually contains shell commands that form a shutdown procedure. For example, one common logout command is the one to check for any active background jobs; another is to clear the screen and then issue a farewell message.

As with **.login**, you can add your own shell commands to the **.logout** file. Using the Vi editor, you can change the farewell message or add other operations. In the next example, the user has a `clear` and an `echo` command in the **.logout** file. When the user logs out, the `clear` command will clear the screen, and `echo` will display the message "Good-bye for now".

.logout

```
clear
echo "Good-bye for now"
```

CHAPTER 6

Linux Files, Directories, and Archives

In Linux, all files are organized into directories that, in turn, are hierarchically connected to each other in one overall file structure. A file is referenced not according to just its name, but also according to its place in this file structure. You can create as many new directories as you want, adding more directories to the file structure. The Linux file commands can perform sophisticated operations, such as moving or copying whole directories along with their subdirectories. You can use file operations such as **`find`**, **`cp`**, **`mv`**, and **`ln`** to locate files and copy, move, or link them from one directory to another. Desktop file managers, such as Konqueror and Nautilus used on the KDE and GNOME desktops, provide a graphical user interface to perform the same operations using icons, windows, and menus (see Chapters 8 and 9). This chapter will focus on the commands you use in the shell command line to manage files, such as **`cp`** and **`mv`**. However, whether you use the command line or a GUI file manager, the underlying file structure is the same.

The organization of the Linux file structure into its various system and network administration directories is discussed in detail in Chapter 32. Though not part of the Linux file structure, there are also special tools you can use to access Windows partitions and floppy disks. These follow much the same format as Linux file commands.

Archives are used to back up files or to combine them into a package, which can then be transferred as one file over the Internet or posted on an FTP site for easy downloading. The standard archive utility used on Linux and Unix systems is tar, for which several GUI front ends exist. You have several compression programs to choose from, including GNU zip (gzip), Zip, bzip, and compress.

NOTE *Linux also allows you to mount and access file systems used by other operating systems such as Unix or Windows. Linux itself supports a variety of different file systems such as ext2, ext3, and ReiserFS.*

Linux Files

You can name a file using any letters, underscores, and numbers. You can also include periods and commas. Except in certain special cases, you should never begin a filename with a period. Other characters, such as slashes, question marks, or asterisks, are reserved for use as special characters by the system and should not be part of a filename. Filenames can be as long as 256 characters. Filenames can also include spaces, though to reference such filenames from the command line, be sure to encase them in quotes. On a desktop like GNOME or KDE, you do not need quotes.

You can include an extension as part of a filename. A period is used to distinguish the filename proper from the extension. Extensions can be useful for categorizing your files. You are probably familiar with certain standard extensions that have been adopted by convention. For example, C source code files always have an extension of **.c**. Files that contain compiled object code have a **.o** extension. You can, of course, make up your own file extensions. The following examples are all valid Linux filenames. Keep in mind that to reference the last of these names on the command line, you would have to encase it in quotes as "New book review":

```
preface
chapter2
9700info
New_Revisions
calc.c
intro.bk1
New book review
```

Special initialization files are also used to hold shell configuration commands. These are the hidden, or dot, files, which begin with a period. Dot files used by commands and applications have predetermined names, such as the **.mozilla** directory used to hold your Mozilla data and configuration files. Recall that when you use **ls** to display your filenames, the dot files will not be displayed. To include the dot files, you need to use **ls** with the **-a** option. Dot files are discussed in more detail in Chapter 5.

The **ls -l** command displays detailed information about a file. First the permissions are displayed, followed by the number of links, the owner of the file, the name of the group the user belongs to, the file size in bytes, the date and time the file was last modified, and the name of the file. Permissions indicate who can access the file: the user, members of a group, or all other users. Permissions are discussed in detail later in this chapter. The group name indicates the group permitted to access the file object. In the example in the next paragraph, the file type for **mydata** is that of an ordinary file. Only one link exists, indicating the file has no other names and no other links. The owner's name is **chris**, the same as the login name, and the group name is **weather**. Other users probably also belong to the **weather** group. The size of the file is 207 bytes, and it was last modified on February 20 at 11:55 A.M. The name of the file is **mydata**.

If you want to display this detailed information for all the files in a directory, simply use the **ls -l** command without an argument.

```
$ ls -l
-rw-r--r-- 1 chris weather 207 Feb 20 11:55 mydata
-rw-rw-r-- 1 chris weather 568 Feb 14 10:30 today
-rw-rw-r-- 1 chris weather 308 Feb 17 12:40 monday
```

All files in Linux have one physical format—a byte stream. A *byte stream* is just a sequence of bytes. This allows Linux to apply the file concept to every data component in the system. Directories are classified as files, as are devices. Treating everything as a file allows Linux to organize and exchange data more easily. The data in a file can be sent directly to a device such as a screen because a device interfaces with the system using the same byte-stream file format as regular files.

This same file format is used to implement other operating system components. The interface to a device, such as the screen or keyboard, is designated as a file. Other components, such as directories, are themselves byte-stream files, but they have a special internal organization. A directory file contains information about a directory, organized in a special directory format. Because these different components are treated as files, they can be said to constitute different *file types*. A character device is one file type. A directory is another file type. The number of these file types may vary according to your specific implementation of Linux. Five common types of files exist, however: ordinary files, directory files, first-in first-out pipes, character device files, and block device files. Although you may rarely reference a file's type, it can be useful when searching for directories or devices. Later in the chapter, you'll see how to use the file type in a search criterion with the **`find`** command to search specifically for directory or device names.

Although all ordinary files have a byte-stream format, they may be used in different ways. The most significant difference is between binary and text files. Compiled programs are examples of binary files. However, even text files can be classified according to their different uses. You can have files that contain C programming source code or shell commands, or even a file that is empty. The file could be an executable program or a directory file. The Linux **`file`** command helps you determine what a file is used for. It examines the first few lines of a file and tries to determine a classification for it. The **`file`** command looks for special keywords or special numbers in those first few lines, but it is not always accurate. In the next example, the **`file`** command examines the contents of two files and determines a classification for them:

```
$ file monday reports
monday: text
reports: directory
```

If you need to examine the entire file byte by byte, you can do so with the **`od`** (octal dump) command. The **`od`** command performs a dump of a file. By default, it prints every byte in its octal representation. However, you can also specify a character, decimal, or hexadecimal representation. The **`od`** command is helpful when you need to detect any special character in your file or if you want to display a binary file.

The File Structure

Linux organizes files into a hierarchically connected set of directories. Each directory may contain either files or other directories. In this respect, directories perform two important functions. A *directory* holds files, much like files held in a file drawer, and a directory connects to other directories, much as a branch in a tree is connected to other branches. Because of the similarities to a tree, such a structure is often referred to as a *tree structure*.

The Linux file structure branches into several directories beginning with a root directory, /. Within the root directory, several system directories contain files and programs that are features of the Linux system. The root directory also contains a directory called **/home** that contains the home directories of all the users in the system. Each user's home directory, in turn, contains the directories the user has made for his or her own use. Each of these can also contain directories. Such nested directories branch out from the user's home directory.

NOTE *The user's home directory can be any directory, though it is usually the directory that bears the user's login name. This directory is located in the directory named* ***/home*** *on your Linux system. For example, a user named* ***dylan*** *will have a home directory called* ***dylan*** *located in the system's* ***/home*** *directory. The user's home directory is a subdirectory of the directory called* ***/home*** *on your system.*

Home Directories

When you log in to the system, you are placed within your home directory. The name given to this directory by the system is the same as your login name. Any files you create when you first log in are organized within your home directory. Within your home directory, however, you can create more directories. You can then change to these directories and store files in them. The same is true for other users on the system. Each user has his or her own home directory, identified by the appropriate login name. Users, in turn, can create their own directories.

You can access a directory either through its name or by making it your working directory. Each directory is given a name when it is created. You can use this name in file operations to access files in that directory. You can also make the directory your working directory. If you do not use any directory names in a file operation, the working directory will be accessed. The working directory is the one from which you are currently working. When you log in, the working directory is your home directory, usually having the same name as your login name. You can change the working directory by using the **`cd`** command to designate another directory as the working directory.

Pathnames

The name you give to a directory or file when you create it is not its full name. The full name of a directory is its *pathname.* The hierarchically nested relationship among directories forms paths, and these paths can be used to identify and reference any directory or file uniquely or absolutely. Each directory in the file structure can be said to have its own unique path. The actual name by which the system identifies a directory always begins with the root directory and consists of all directories nested below that directory.

In Linux, you write a pathname by listing each directory in the path separated from the last by a forward slash. A slash preceding the first directory in the path represents the root. The pathname for the **robert** directory is **/home/robert**. The pathname for the **reports** directory is **/home/chris/reports**. Pathnames also apply to files. When you create a file within a directory, you give the file a name. The actual name by which the system identifies the file, however, is the filename combined with the path of directories from the root to the file's directory. As an example, the pathname for **monday** is **/home/chris/reports/monday** (the root directory is represented by the first slash). The path for the **monday** file consists of the root, **home**, **chris**, and **reports** directories and the filename **monday**.

Pathnames may be absolute or relative. An *absolute pathname* is the complete pathname of a file or directory beginning with the root directory. A *relative pathname* begins from your working directory; it is the path of a file relative to your working directory. The working directory is the one you are currently operating in. Using the previous example, if **chris** is your working directory, the relative pathname for the file **monday** is **reports/monday**. The absolute pathname for **monday** is **/home/chris/reports/monday**.

The absolute pathname from the root to your home directory can be especially complex and, at times, even subject to change by the system administrator. To make it easier to reference, you can use a special character, the tilde (**~**), which represents the absolute pathname of your home directory. In the next example, from the **thankyou** directory, the user references the **weather** file in the home directory by placing a tilde and slash before **weather**:

```
$ pwd
/home/chris/letters/thankyou
$ cat ~/weather
raining and warm
$
```

You must specify the rest of the path from your home directory. In the next example, the user references the **monday** file in the **reports** directory. The tilde represents the path to the user's home directory, **/home/chris**, and then the rest of the path to the **monday** file is specified.

```
$ cat ~/reports/monday
```

System Directories

The root directory that begins the Linux file structure contains several system directories. The system directories contain files and programs used to run and maintain the system. Many contain other subdirectories with programs for executing specific features of Linux. For example, the directory **/usr/bin** contains the various Linux commands that users execute, such as **lp**l. The directory **/bin** holds system-level commands. Table 6-1 lists the basic system directories.

Listing, Displaying, and Printing Files: ls, cat, more, less, and lpr

One of the primary functions of an operating system is the management of files. You may need to perform certain basic output operations on your files, such as displaying them on your screen or printing them. The Linux system provides a set of commands that perform basic file-management operations, such as listing, displaying, and printing files, as well as copying, renaming, and erasing files. The command names are usually made up of abbreviated versions of words. For example, the **ls** command is a shortened form of "list" and lists the files in your directory. The **lpr** command is an abbreviated form of "line print" and will print a file. The **cat**, **less**, and **more** commands display the contents of a file on the screen. Table 6-2 lists these commands with their different options. When you log in to your Linux system, you may want a list of the files in your home directory. The **ls** command, which outputs a list of your file and directory names, is useful for this. The **ls** command has many possible options for displaying filenames according to specific features.

Directory	Function
/	Begins the file system structure, called the *root*.
/home	Contains users' home directories.
/bin	Holds all the standard commands and utility programs.
/usr	Holds those files and commands used by the system; this directory breaks down into several subdirectories.
/usr/bin	Holds user-oriented commands and utility programs.
/usr/sbin	Holds system administration commands.
/usr/lib	Holds libraries for programming languages.
/usr/share/doc	Holds Linux documentation.
/usr/share/man	Holds the online Man files.
/var/spool	Holds spooled files, such as those generated for printing jobs and network transfers.
/sbin	Holds system administration commands for booting the system.
/var	Holds files that vary, such as mailbox files.
/dev	Holds file interfaces for devices such as the terminals and printers (dynamically generated by udev, do not edit).
/etc	Holds system configuration files and any other system files.

TABLE 6-1 Standard System Directories in Linux

Displaying Files: cat, less, and more

You may also need to look at the contents of a file. The **cat** and **more** commands display the contents of a file on the screen. The name **cat** stands for *concatenate.*

```
$ cat mydata
computers
```

The **cat** command outputs the entire text of a file to the screen at once. This presents a problem when the file is large because its text quickly speeds past on the screen. The **more** and **less** commands are designed to overcome this limitation by displaying one screen of text at a time. You can then move forward or backward in the text at your leisure. You invoke the **more** or **less** command by entering the command name followed by the name of the file you want to view (**less** is a more powerful and configurable display utility).

```
$ less mydata
```

When **more** or **less** invokes a file, the first screen of text is displayed. To continue to the next screen, you press the F key or the SPACEBAR. To move back in the text, you press the B key. You can quit at any time by pressing the Q key.

Command	Execution
ls	Lists file and directory names.
cat *filenames*	Displays a file. It can take filenames for its arguments. It outputs the contents of those files directly to the standard output, which, by default, is directed to the screen.
more *filenames*	Displays a file screen by screen. Press the SPACEBAR to continue to the next screen and q to quit.
less *filenames*	Displays a file screen by screen. Press the SPACEBAR to continue to the next screen and q to quit.
lpr *filenames*	Sends a file to the line printer to be printed; a list of files may be used as arguments. Use the **-P** option to specify a printer.
lpq	Lists the print queue for printing jobs.
lprm	Removes a printing job from the print queue.

TABLE 6-2 Listing, Displaying, and Printing Files

Printing Files: lpr, lpq, and lprm

With the printer commands such as **lpr** and **lprm**, you can perform printing operations such as printing files or canceling print jobs (see Table 6-2). When you need to print files, use the **lpr** command to send files to the printer connected to your system. In the next example, the user prints the **mydata** file:

```
$ lpr mydata
```

If you want to print several files at once, you can specify more than one file on the command line after the **lpr** command. In the next example, the user prints out both the **mydata** and **preface** files:

```
$ lpr mydata preface
```

Printing jobs are placed in a queue and printed one at a time in the background. You can continue with other work as your files print. You can see the position of a particular printing job at any given time with the **lpq** command, which gives the owner of the printing job (the login name of the user who sent the job), the print job ID, the size in bytes, and the temporary file in which it is currently held.

If you need to cancel an unwanted printing job, you can do so with the **lprm** command, which takes as its argument either the ID number of the printing job or the owner's name. It then removes the print job from the print queue. For this task, **lpq** is helpful, for it provides you with the ID number and owner of the printing job you need to use with **lprm**.

Managing Directories: mkdir, rmdir, ls, cd, and pwd

You can create and remove your own directories, as well as change your working directory, with the **mkdir**, **rmdir**, and **cd** commands. Each of these commands can take as its argument the pathname for a directory. The **pwd** command displays the absolute pathname of your

working directory. In addition to these commands, the special characters represented by a single dot, a double dot, and a tilde can be used to reference the working directory, the parent of the working directory, and the home directory, respectively. Taken together, these commands enable you to manage your directories. You can create nested directories, move from one directory to another, and use pathnames to reference any of your directories. Those commands commonly used to manage directories are listed in Table 6-3.

Creating and Deleting Directories

You create and remove directories with the **mkdir** and **rmdir** commands. In either case, you can also use pathnames for the directories. In the next example, the user creates the directory **reports**. Then the user creates the directory **letters** using a pathname:

```
$ mkdir reports
$ mkdir /home/chris/letters
```

Command	Execution
mkdir *directory*	Creates a directory.
rmdir *directory*	Erases a directory.
ls -F	Lists directory name with a preceding slash.
ls -R	Lists working directory as well as all subdirectories.
cd *directory name*	Changes to the specified directory, making it the working directory. **cd** without a directory name changes back to the home directory: **$ cd reports**
pwd	Displays the pathname of the working directory.
directory name/filename	A slash is used in pathnames to separate each directory name. In the case of pathnames for files, a slash separates the preceding directory names from the filename.
..	References the parent directory. You can use it as an argument or as part of a pathname: **$ cd ..** **$ mv ../larisa oldletters**
.	References the working directory. You can use it as an argument or as part of a pathname: **$ ls .**
~/pathname	The tilde is a special character that represents the pathname for the home directory. It is useful when you need to use an absolute pathname for a file or directory: **$ cp monday ~/today**

Table 6-3 Directory Commands

You can remove a directory with the **rmdir** command followed by the directory name. In the next example, the user removes the directory **reports** with the **rmdir** command:

```
$ rmdir reports
```

To remove a directory and all its subdirectories, you use the **rm** command with the **-r** option. This is a very powerful command and can easily be used to erase all your files. If your **rm** command is aliased as **rm -i** (interactive mode), you will be prompted for each file. To simply remove all files and subdirectories without prompts, add the **-f** option. The following example deletes the **reports** directory and all its subdirectories:

```
rm -rf reports
```


Displaying Directory Contents

You have seen how to use the **ls** command to list the files and directories within your working directory. To distinguish between file and directory names, however, you need to use the **ls** command with the **-F** option. A slash is then placed after each directory name in the list.

```
$ ls
weather reports letters
$ ls -F
weather reports/ letters/
```

The **ls** command also takes as an argument any directory name or directory pathname. This enables you to list the files in any directory without first having to change to that directory. In the next example, the **ls** command takes as its argument the name of a directory, **reports**. Then the **ls** command is executed again, only this time the absolute pathname of **reports** is used.

```
$ ls reports
monday tuesday
$ ls /home/chris/reports
monday tuesday
$
```

Moving Through Directories

The **cd** command takes as its argument the name of the directory to which you want to change. The name of the directory can be the name of a subdirectory in your working directory or the full pathname of any directory on the system. If you want to change back to your home directory, you only need to enter the **cd** command by itself, without a filename argument.

```
$ cd props
$ pwd
/home/dylan/props
```

Referencing the Parent Directory

A directory always has a parent (except, of course, for the root). For example, in the preceding listing, the parent for **props** is the **dylan** directory. When a directory is created, two entries are made: one represented with a dot (**.**), and the other with double dots (**..**). The dot represents the pathname of the directory, and the double dots represent the pathname of its parent directory. Double dots, used as an argument in a command, reference a parent directory. The single dot references the directory itself.

You can use the single dot to reference your working directory, instead of using its pathname. For example, to copy a file to the working directory, retaining the same name, the dot can be used in place of the working directory's pathname. In this sense, the dot is another name for the working directory. In the next example, the user copies the **weather** file from the **chris** directory to the **reports** directory. The **reports** directory is the working directory and can be represented with the single dot.

```
$ cd reports
$ cp /home/chris/weather .
```

The **..** symbol is often used to reference files in the parent directory. In the next example, the **cat** command displays the **weather** file in the parent directory. The pathname for the file is the **..** symbol followed by a slash and the filename.

```
$ cat ../weather
raining and warm
```

TIP *You can use the* **cd** *command with the* **..** *symbol to step back through successive parent directories of the directory tree from a lower directory.*

File and Directory Operations: find, cp, mv, rm, and ln

As you create more and more files, you may want to back them up, change their names, erase some of them, or even give them added names. Linux provides you with several file commands that enable you to search for files, copy files, rename files, or remove files (see Table 6-5 later in this chapter). If you have a large number of files, you can also search them to locate a specific one. The command names shortened forms of full words, consisting of only two characters. The **cp** command stands for "copy" and copies a file, **mv** stands for "move" and renames or moves a file, **rm** stands for "remove" and erases a file, and **ln** stands for "link" and adds another name for a file, often used as a shortcut to the original. One exception to the two-character rule is the **find** command, which performs searches of your filenames to find a file. All these operations can be handled by the GUI desktops such as GNOME and KDE (see Chapters 7 and 8).

Searching Directories: find

Once you have a large number of files in many different directories, you may need to search them to locate a specific file, or files, of a certain type. The **find** command enables you to perform such a search from the command line. The **find** command takes as its arguments directory names followed by several possible options that specify the type of search and the

criteria for the search; it then searches within the directories listed and their subdirectories for files that meet these criteria. The **find** command can search for a file by name, type, owner, and even the time of the last update.

```
$ find directory-list -option criteria
```

TIP *From the GNOME desktop you can use the Search tool in the Places menu to search for files. From the KDE Desktop you can use the find tool in the file manager. Select Find from the file manager (Konqueror) tools menu.*

The **-name** option has as its criteria a pattern and instructs **find** to search for the filename that matches that pattern. To search for a file by name, you use the **find** command with the directory name followed by the **-name** option and the name of the file.

```
$ find directory-list -name filename
```

The **find** command also has options that merely perform actions, such as outputting the results of a search. If you want **find** to display the filenames it has found, you simply include the **-print** option on the command line along with any other options. The **-print** option is an action that instructs **find** to write to the standard output the names of all the files it locates (you can also use the **-ls** option instead to list files in the long format). In the next example, the user searches for all the files in the **reports** directory with the name **monday**. Once located, the file, with its relative pathname, is printed.

```
$ find reports -name monday -print
reports/monday
```

The **find** command prints out the filenames using the directory name specified in the directory list. If you specify an absolute pathname, the absolute path of the found directories will be output. If you specify a relative pathname, only the relative pathname will be output. In the preceding example, the user specified a relative pathname, **reports**, in the directory list. Located filenames were output beginning with this relative pathname. In the next example, the user specifies an absolute pathname in the directory list. Located filenames are then output using this absolute pathname.

```
$ find /home/chris -name monday -print
/home/chris/reports/monday
```

TIP *Should you need to find the location of a specific program or configuration file, you can use* ***find*** *to search for the file from the root directory. Log in as the root user and use / as the directory. This command searches for the location of the* ***more*** *command and files on the entire file system:* ***find / -name more -print****.*

Searching the Working Directory

If you want to search your working directory, you can use the dot in the directory pathname to represent your working directory. The double dots represent the parent directory. The next example searches all files and subdirectories in the working directory, using the dot to represent the working directory. If you are located in your home directory, this is a convenient

way to search through all your own directories. Notice the located filenames are output beginning with a dot.

```
$ find . -name weather -print
./weather
```

You can use shell wildcard characters as part of the pattern criterion for searching files. The special character must be quoted, however, to avoid evaluation by the shell. In the next example, all files with the **.c** extension in the **programs** directory are searched for and then displayed in the long format using the **-ls** action:

```
$ find programs -name '*.c' -ls
```

Locating Directories

You can also use the **find** command to locate other directories. In Linux, a directory is officially classified as a special type of file. Although all files have a byte-stream format, some files, such as directories, are used in special ways. In this sense, a file can be said to have a file type. The **find** command has an option called **-type** that searches for a file of a given type. The **-type** option takes a one-character modifier that represents the file type. The modifier that represents a directory is a **d**. In the next example, both the directory name and the directory file type are used to search for the directory called **thankyou**:

```
$ find /home/chris -name thankyou -type d -print
/home/chris/letters/thankyou
$
```

File types are not so much different types of files as they are the file format applied to other components of the operating system, such as devices. In this sense, a device is treated as a type of file, and you can use **find** to search for devices and directories, as well as ordinary files. Table 6-4 lists the different types available for the **find** command's **-type** option.

You can also use the find operation to search for files by ownership or security criteria, like those belonging to a specific user or those with a certain security context. The **user** option lets you locate all files belonging to a certain user. The following example lists all files that the user **chris** has created or owns on the entire system. To list those just in the users' home directories, you use **/home** for the starting search directory. This finds all files in a user's home directory as well as any owned by that user in other user directories.

```
$ find / -user chris -print
```

Copying Files

To make a copy of a file, you simply give **cp** two filenames as its arguments (see Table 6-5). The first filename is the name of the file to be copied—the one that already exists. This is often referred to as the *source file.* The second filename is the name you want for the copy. This will be a new file containing a copy of all the data in the source file. This second argument is often referred to as the *destination file.* The syntax for the **cp** command follows:

```
$ cp source-file destination-file
```

Command or Option	Execution
find	Searches directories for files according to search criteria. This command has several options that specify search criteria and actions to be taken.
-name *pattern*	Searches for files with *pattern* in the name.
-lname *pattern*	Searches for symbolic link files.
-group *name*	Searches for files belonging to the group *name.*
-gid *name*	Searches for files belonging to a group according to group ID.
-user *name*	Searches for files belonging to a user.
-uid *name*	Searches for files belonging to a user according to user ID.
-size *num***c**	Searches for files with the size *num* in blocks. If **c** is added after *num,* the size in bytes (characters) is searched for.
-mtime *num*	Searches for files last modified *num* days ago.
-newer *pattern*	Searches for files modified after the one matched by *pattern.*
-context *scontext*	Searches for files according to security context (SE Linux).
-print	Outputs the result of the search to the standard output. The result is usually a list of filenames, including their full pathnames.
-type *filetype*	Searches for files with the specified file type. File type can be **b** for block device, **c** for character device, **d** for directory, **f** for file, or **l** for symbolic link.
-perm *permission*	Searches for files with certain permissions set. Use octal or symbolic format for permissions.
-ls	Provides a detailed listing of each file, with owner, permission, size, and date information.
-exec *command*	Executes command when files found.

TABLE 6-4 The **find** Command

In the next example, the user copies a file called **proposal** to a new file called **oldprop**:

```
$ cp proposal oldprop
```

You can unintentionally destroy another file with the **cp** command. The **cp** command generates a copy by first creating a file and then copying data into it. If another file has the same name as the destination file, that file will be destroyed and a new file with that name created. By default Red Hat configures your system to check for an existing copy by the same name (**cp** is aliased with the **-i** option; see Chapter 5). To copy a file from your working directory to another directory, you only need to use that directory name as the

Command	Execution
cp *filename filename*	Copies a file. **cp** takes two arguments: the original file and the name of the new copy. You can use pathnames for the files to copy across directories: **$ cp today reports/Monday**
cp -r *dirname dirname*	Copies a subdirectory from one directory to another. The copied directory includes all its own subdirectories: **$ cp -r letters/thankyou oldletters**
mv *filename filename*	Moves (renames) a file. The **mv** command takes two arguments: the first is the file to be moved. The second argument can be the new filename or the pathname of a directory. If it is the name of a directory, then the file is literally moved to that directory, changing the file's pathname: **$ mv today /home/chris/reports**
mv *dirname dirname*	Moves directories. In this case, the first and last arguments are directories: **$ mv letters/thankyou oldletters**
ln *filename filename*	Creates added names for files referred to as links. A link can be created in one directory that references a file in another directory: **$ ln today reports/Monday**
rm *filenames*	Removes (erases) a file. Can take any number of filenames as its arguments. Removes links to a file. If a file has more than one link, you need to remove all of them to erase a file: **$rm today weather weekend**

TABLE 6-5 File Operations

second argument in the **cp** command. In the next example, the **proposal** file is overwritten by the **newprop** file. The **proposal** file already exists.

```
$ cp newprop proposal
```

You can use any of the wildcard characters to generate a list of filenames to use with **cp** or **mv**. For example, suppose you need to copy all your C source code files to a given directory. Instead of listing each one individually on the command line, you can use an * character with the **.c** extension to match and generate a list of C source code files (all files with a **.c** extension). In the next example, the user copies all source code files in the current directory to the **sourcebks** directory:

```
$ cp *.c sourcebks
```

If you want to copy all the files in a given directory to another directory, you can use * to generate a list of all those files in a **cp** command. In the next example, the user copies all the files in the **props** directory to the **oldprop** directory. Notice the use of the **props** pathname preceding the * special characters. In this context, **props** is a pathname that will be appended before each file in the list that * generates.

```
$ cp props/* oldprop
```

You can, of course, use any of the other special characters, such as **.**, **?**, or **[]**. In the next example, the user copies both source code and object code files (**.c** and **.o**) to the **projbk** directory:

```
$ cp *.[oc] projbk
```

When you copy a file, you may want to give the copy a different name than the original. To do so, place the new filename after the directory name, separated by a slash.

```
$ cp filename directory-name/new-filename
```


Moving Files

You can use the **mv** command to either rename a file or move a file from one directory to another. When using **mv** to rename a file, you simply use the new filename as the second argument. The first argument is the current name of the file you are renaming. If you want to rename a file when you move it, you can specify the new name of the file after the directory name. In the next example, the **proposal** file is renamed with the name **version1**:

```
$ mv proposal version1
```

As with **cp**, it is easy for **mv** to erase a file accidentally. When renaming a file, you might accidentally choose a filename already used by another file. In this case, that other file will be erased. The **mv** command also has an **-i** option that checks first to see if a file by that name already exists.

You can also use any of the special characters described in Chapter 3 to generate a list of filenames to use with **mv**. In the next example, the user moves all C source code files in the current directory to the **newproj** directory:

```
$ mv *.c newproj
```

If you want to move all the files in a given directory to another directory, you can use ***** to generate a list of all those files. In the next example, the user moves all the files in the **reports** directory to the **repbks** directory:

```
$ mv reports/* repbks
```

NOTE *On GNOME or KDE, the easiest way to copy files to a CD-R/RW or DVD-R/RW disc is to use the built-in desktop burning capability. Just insert a blank disk, open it as a folder, and drag and drop files on to it. You will be prompted automatically to burn the files. You can also use any number or CD/DVD burning tools, such as K3B.*

Copying and Moving Directories

You can also copy or move whole directories at once. Both **cp** and **mv** can take as their first argument a directory name, enabling you to copy or move subdirectories from one directory into another (see Table 6-5). The first argument is the name of the directory to be moved or copied, while the second argument is the name of the directory within which it is to be placed. The same pathname structure used for files applies to moving or copying directories.

You can just as easily copy subdirectories from one directory to another. To copy a directory, the **cp** command requires you to use the **-r** option. The **-r** option stands for "recursive." It directs the **cp** command to copy a directory, as well as any subdirectories it may contain. In other words, the entire directory subtree, from that directory on, will be copied. In the next example, the **thankyou** directory is copied to the **oldletters** directory. Now two **thankyou** subdirectories exist, one in **letters** and one in **oldletters**.

```
$ cp -r letters/thankyou oldletters
$ ls -F letters
/thankyou
$ ls -F oldletters
/thankyou
```

Erasing Files and Directories: The rm Command

As you use Linux, you will find the number of files you use increases rapidly. Generating files in Linux is easy. Applications such as editors, and commands such as **cp**, easily create files. Eventually, many of these files may become outdated and useless. You can then remove them with the **rm** command. The **rm** command can take any number of arguments, enabling you to list several filenames and erase them all at the same time. In the next example, the user erases the file **oldprop**:

```
$ rm oldprop
```

Be careful when using the **rm** command, because it is irrevocable. Once a file is removed, it cannot be restored (there is no undo). With the **-i** option, you are prompted separately for each file and asked whether to remove it. If you enter **y**, the file will be removed. If you enter anything else, the file is not removed. In the next example, the **rm** command is instructed to erase the files **proposal** and **oldprop**. The **rm** command then asks for confirmation for each file. The user decides to remove **oldprop,** but not **proposal**.

```
$ rm -i proposal oldprop
Remove proposal? n
Remove oldprop? y
$
```

Links: The ln Command

You can give a file more than one name using the **ln** command. You might want to reference a file using different filenames to access it from different directories. The added names are often referred to as *links*. Linux supports two different types of links, hard and symbolic. *Hard* links are literally another name for the same file, whereas *symbolic* links function like shortcuts referencing another file. Symbolic links are much more flexible and can work over many different file systems, whereas hard links are limited to your local file system. Furthermore, hard links introduce security concerns, as they allow direct access from a link that may have public access to an original file that you may want protected. Because of this, links are usually implemented as symbolic links.

Symbolic Links

To set up a symbolic link, you use the **ln** command with the **-s** option and two arguments: the name of the original file and the new, added filename. The **ls** operation lists both filenames, but only one physical file will exist.

```
$ ln -s original-file-name added-file-name
```

In the next example, the **today** file is given the additional name **weather**. It is just another name for the **today** file.

```
$ ls
today
$ ln -s today weather
$ ls
today weather
```

You can give the same file several names by using the **ln** command on the same file many times. In the next example, the file **today** is given both the names **weather** and **weekend**:

```
$ ln -s today weather
$ ln -s today weekend
$ ls
today weather weekend
```

If you list the full information about a symbolic link and its file, you will find the information displayed is different. In the next example, the user lists the full information for both **lunch** and **/home/george/veglist** using the **ls** command with the **-l** option. The first character in the line specifies the file type. Symbolic links have their own file type, represented by an **l**. The file type for **lunch** is **l**, indicating it is a symbolic link, not an ordinary file. The number after the term "group" is the size of the file. Notice the sizes differ. The size of the **lunch** file is only four bytes. This is because **lunch** is only a symbolic link—a file that holds the pathname of another file—and a pathname takes up only a few bytes. It is not a direct hard link to the **veglist** file.

```
$ ls -l lunch /home/george/veglist
-rw-rw-r-- 1 george group 793 Feb 14 10:30 veglist
lrw-rw-r-- 1 chris group 4 Feb 14 10:30 lunch
```

To erase a file, you need to remove only its original name (and any hard links to it). If any symbolic links are left over, they will be unable to access the file. In this case, a symbolic link will hold the pathname of a file that no longer exists.

Hard Links

You can give the same file several names by using the **ln** command on the same file many times. To set up a hard link, you use the **ln** command with no **-s** option and two arguments: the name of the original file and the new, added filename. The **ls** operation lists both filenames, but only one physical file will exist.

```
$ ln original-file-name added-file-name
```

In the next example, the **monday** file is given the additional name **storm**. It is just another name for the **monday** file.

```
$ ls
today
$ ln monday storm
$ ls
monday storm
```

To erase a file that has hard links, you need to remove all its hard links. The name of a file is actually considered a link to that file—hence the command **rm** that removes the link to the file. If you have several links to the file and remove only one of them, the others stay in place and you can reference the file through them. The same is true even if you remove the original link—the original name of the file. Any added links will work just as well. In the next example, the **today** file is removed with the **rm** command. However, a link to that same file exists, called **weather**. The file can then be referenced under the name **weather**.

```
$ ln today weather
$ rm today
$ cat weather
The storm broke today
and the sun came out.
$
```

NOTE *Each file and directory in Linux contains a set of permissions that determine who can access them and how. You set these permissions to limit access in one of three ways: you can restrict access to yourself alone, you can allow users in a group to have access, or you can permit anyone on your system to have access. You can also control how a given file or directory is accessed. A file and directory may have read, write, and execute permissions. When a file is created, it is automatically given read and write permissions for the owner, enabling you to display and modify the file. You may change these permissions to any combination you want (see Chapter 28 for more details).*

The mtools Utilities: msdos

Your Linux system provides a set of utilities, known as *mtools,* that enable you to easily access floppy and hard disks formatted for MS-DOS. They work only with the old MS-DOS or FAT32 file systems, not with Windows Vista, XP, NT, or 2000, which use the NTFS file system. The **mcopy** command enables you to copy files to and from an MS-DOS floppy disk in your floppy drive or a Windows FAT32 partition on your hard drive. No special operations, such as mounting, are required. With mtools, you needn't mount an MS-DOS partition to access it. For an MS-DOS floppy disk, once you place the disk in your floppy drive, you can use mtool commands to access those files. For example, to copy a file from an MS-DOS floppy disk to your Linux system, use the **mcopy** command. You specify the MS-DOS disk with **a:** for the A drive. Unlike normal DOS pathnames, pathnames used with mtool commands use forward slashes instead of backslashes. The directory **docs** on the A drive would be referenced by the pathname **a:/docs**, not **a:\docs**. Unlike MS-DOS, which defaults the second argument to the current directory, you always need to supply the second argument for **mcopy**. The next

example copies the file **mydata** to the MS-DOS disk and then copies the **preface** file from the disk to the current Linux directory.

```
$ mcopy mydata a:
$ mcopy a:/preface  .
```

TIP *You can use mtools to copy data to Windows-formatted floppy disks or to a Windows FAT32 partition, which can also be read or written to by Windows XP, but you cannot access Windows Vista, XP, NT, or 2000 hard disk file systems (NTFS) with mtools. The NTFS partitions require a different tool, the NTFS kernel module.*

You can use the **mdir** command to list files on your MS-DOS disk, and you can use the **mcd** command to change directories on it. The next example lists the files on the MS-DOS disk in your floppy drive and then changes to the **docs** directory on that drive:

```
$ mdir a:
$ mcd a:/docs
```

Access to MS-DOS or Windows 95, 98, or Me partitions by mtools is configured by the **/etc/mtools.conf** file. This file lists several different default MS-DOS or Windows partitions and disk drives. Each drive or partition is identified with a particular device name.

Archiving and Compressing Files

Archives are used to back up files or to combine them into a package, which can then be transferred as one file over the Internet or posted on an FTP site for easy downloading. The standard archive utility used on Linux and Unix systems is tar, for which several GUI front ends exist. You have several compression programs to choose from, including GNU zip (gzip), Zip, bzip, and compress.

TIP *You can use the unrar tool to read and extract the popular rar archives but not to create them. unrar is available from* ***rpm.livna.org*** *and can be downloaded and installed with yum. File Roller is able to extract RAR files once the unrar tool is installed. Other graphical front ends, such as Xarchiver and Linrar, are available from* ***freshmeat.net.*** *To create rar archives, you have to purchase the archiver from Rarlab at* ***rarlab.com.***

Archiving and Compressing Files with File Roller

GNOME provides the File Roller tool (accessible from the Accessories menu, labeled Archive Manager) that operates as a GUI front end to archive and compress files, letting you perform Zip, gzip, tar, and bzip2 operations using a GUI. You can examine the contents of archives, extract the files you want, and create new compressed archives. When you create an archive, you determine its compression method by specifying its filename extension, such as **.gz** for gzip or **.bz2** for bzip2. You can select the different extensions from the File Type menu or enter the extension yourself. To both archive and compress files, you can choose a combined extension like **.tar.bz2**, which both archives with tar and compresses with bzip2.

Click Add to add files to your archive. To extract files from an archive, open the archive to display the list of archive files. You can then click Extract to extract particular files or the entire archive.

TIP *File Roller can also be used to examine the contents of an archive file easily. From the file manager, right-click the archive and select Open With Archive Manager. The list of files and directories in that archive will be displayed. For subdirectories, double-click their entries. This method also works for RPM software files, letting you browse all the files that make up a software package.*

Archive Files and Devices: tar

The tar utility creates archives for files and directories. With tar, you can archive specific files, update them in the archive, and add new files to an archive. You can even archive entire directories with all their files and subdirectories, all of which can be restored from the archive. The tar utility was originally designed to create archives on tapes. (The term "tar" stands for tape archive. However, you can create archives on any device, such as a floppy disk, or you can create an archive file to hold the archive.) The tar utility is ideal for making backups of your files or combining several files into a single file for transmission across a network (File Roller is a GUI for tar).

NOTE *As an alternative to tar, you can use pax, which is designed to work with different kinds of Unix archive formats such as cpio, bcpio, and tar. You can extract, list, and create archives. The pax utility is helpful if you are handling archives created on Unix systems that are using different archive formats.*

Displaying Archive Contents

Both file managers in GNOME and the K Desktop have the capability to display the contents of a tar archive file automatically. The contents are displayed as though they were files in a directory. You can list the files as icons or with details, sorting them by name, type, or other fields. You can even display the contents of files. Clicking a text file opens it with a text editor, and an image is displayed with an image viewer. If the file manager cannot determine what program to use to display the file, it prompts you to select an application. Both file managers can perform the same kinds of operations on archives residing on remote file systems, such as tar archives on FTP sites. You can obtain a listing of their contents and even read their readme files. The Nautilus file manager (GNOME) can also extract an archive. Right-click the Archive icon and select Extract.

Creating Archives

On Linux, tar is often used to create archives on devices or files. You can direct tar to archive files to a specific device or a file by using the **f** option with the name of the device or file. The syntax for the **tar** command using the **f** option is shown in the next example. The device or filename is often referred to as the archive name. When creating a file for a tar archive, the filename is usually given the extension **.tar**. This is a convention only and is not required. You can list as many filenames as you want. If a directory name is specified, all its subdirectories are included in the archive.

```
$ tar optionsf archive-name.tar directory-and-file-names
```

To create an archive, use the **c** option. Combined with the **f** option, **c** creates an archive on a file or device. You enter this option before and right next to the **f** option. Notice no dash precedes a tar option. Table 6-6 lists the different options you can use with tar. In the next example, the directory **mydir** and all its subdirectories are saved in the file **myarch.tar**. In this example, the **mydir** directory holds two files, **mymeeting** and **party**, as well as a directory called **reports** that has three files: **weather**, **monday**, and **friday**.

```
$ tar cvf myarch.tar mydir
mydir/
mydir/reports/
mydir/reports/weather
```


Commands	Execution
tar *options files*	Backs up files to tape, device, or archive file.
tar *optionsf archive_name filelist*	Backs up files to a specific file or device specified as *archive_name. filelist*; can be filenames or directories.
Options	
c	Creates a new archive.
t	Lists the names of files in an archive.
r	Appends files to an archive.
U	Updates an archive with new and changed files; adds only those files modified since they were archived or files not already present in the archive.
--delete	Removes a file from the archive.
w	Waits for a confirmation from the user before archiving each file; enables you to update an archive selectively.
x	Extracts files from an archive.
m	When extracting a file from an archive, no new timestamp is assigned.
M	Creates a multiple-volume archive that may be stored on several floppy disks.
f *archive-name*	Saves the tape archive to the file archive name, instead of to the default tape device. When given an archive name, the **f** option saves the tar archive in a file of that name.
f *device-name*	Saves a tar archive to a device such as a floppy disk or tape. **/dev/fd0** is the device name for your floppy disk; the default device is held in **/etc/default/tar-file**.
v	Displays each filename as it is archived.
z	Compresses or decompresses archived files using gzip.
j	Compresses or decompresses archived files using bzip2.

TABLE 6-6 File Archives: **tar**

```
mydir/reports/monday
mydir/reports/friday
mydir/mymeeting
mydir/party
```

Extracting Archives

The user can later extract files and directories from the archive using the **x** option. The **xf** option extracts files from an archive file or device. The tar extraction operation generates all subdirectories. In the next example, the **xf** option directs **tar** to extract all the files and subdirectories from the tar file **myarch.tar**:

```
$ tar xvf myarch.tar
mydir/
mydir/reports/
mydir/reports/weather
mydir/reports/monday
mydir/reports/friday
mydir/mymeeting
mydir/party
```

You use the **r** option to add files to an already-created archive. The **r** option appends the files to the archive. In the next example, the user appends the files in the **letters** directory to the **myarch.tar** archive. Here, the directory **mydocs** and its files are added to the **myarch.tar** archive:

```
$ tar rvf myarch.tar mydocs
mydocs/
mydocs/doc1
```

Updating Archives

If you change any of the files in your directories you previously archived, you can use the **u** option to instruct tar to update the archive with any modified files. The **tar** command compares the time of the last update for each archived file with those in the user's directory and copies into the archive any files that have been changed since they were last archived. Any newly created files in these directories are also added to the archive. In the next example, the user updates the **myarch.tar** file with any recently modified or newly created files in the **mydir** directory. In this case, the **gifts** file is added to the **mydir** directory.

```
tar uvf myarch.tar mydir
mydir/
mydir/gifts
```

If you need to see what files are stored in an archive, you can use the **tar** command with the **t** option. The next example lists all the files stored in the **myarch.tar** archive:

```
tar tvf myarch.tar
drwxr-xr-x root/root 0 2000-10-24 21:38:18 mydir/
drwxr-xr-x root/root 0 2000-10-24 21:38:51 mydir/reports/
-rw-r--r-- root/root 22 2000-10-24 21:38:40 mydir/reports/weather
-rw-r--r-- root/root 22 2000-10-24 21:38:45 mydir/reports/monday
-rw-r--r-- root/root 22 2000-10-24 21:38:51 mydir/reports/friday
```

```
-rw-r--r-- root/root 22 2000-10-24 21:38:18 mydir/mymeeting
-rw-r--r-- root/root 22 2000-10-24 21:36:42 mydir/party
drwxr-xr-x root/root 0 2000-10-24 21:48:45 mydocs/
-rw-r--r-- root/root 22 2000-10-24 21:48:45 mydocs/doc1
drwxr-xr-x root/root 0 2000-10-24 21:54:03 mydir/
-rw-r--r-- root/root 22 2000-10-24 21:54:03 mydir/gifts
```

Archiving to Floppies

To back up the files to a specific device, specify the device as the archive. For a floppy disk, you can specify the floppy drive. Be sure to use a blank floppy disk. Any data previously placed on it will be erased by this operation. In the next example, the user creates an archive on the floppy disk in the **/dev/fd0** device and copies into it all the files in the **mydir** directory:

```
$ tar cf /dev/fd0 mydir
```

To extract the backed-up files on the disk in the device, use the **xf** option:

```
$ tar xf /dev/fd0
```

Compressing Archives

The **tar** operation does not perform compression on archived files. If you want to compress the archived files, you can instruct tar to invoke the gzip utility to compress them. With the lowercase **z** option, tar first uses gzip to compress files before archiving them. The same **z** option invokes gzip to decompress them when extracting files.

```
$ tar czf myarch.tar.gz mydir
```

To use bzip instead of gzip to compress files before archiving them, you use the **j** option. The same **j** option invokes bzip to decompress them when extracting files.

```
$ tar cjf myarch.tar.bz2 mydir
```

Remember, a difference exists between compressing individual files in an archive and compressing the entire archive as a whole. Often, an archive is created for transferring several files at once as one tar file. To shorten transmission time, the archive should be as small as possible. You can use the compression utility gzip on the archive tar file to compress it, reducing its size, and then send the compressed version. The person receiving it can decompress it, restoring the tar file. Using gzip on a tar file often results in a file with the extension **.tar.gz**. The extension **.gz** is added to a compressed gzip file. The next example creates a compressed version of **myarch.tar** using the same name with the extension **.gz**:

```
$ gzip myarch.tar
$ ls
$ myarch.tar.gz
```

Instead of retyping the **tar** command for different files, you can place the command in a script and pass the files to it. Be sure to make the script executable. In the following example, a simple **myarchprog** script is created that will archive filenames listed as its arguments.

```
myarchprog
tar  cvf   myarch.tar    $*
```

A run of the **myarchprog** script with multiple arguments is shown here:

```
$ myarchprog mydata preface
mydata
preface
```

Archiving to Tape

If you have a default device specified, such as a tape, and you want to create an archive on it, you can simply use **tar** without the **f** option and a device or filename. This can be helpful for making backups of your files. The name of the default device is held in a file called **/etc/default/tar**. The syntax for the **tar** command using the default tape device is shown in the following example. If a directory name is specified, all its subdirectories are included in the archive.

```
$ tar option directory-and-file-names
```

In the next example, the directory **mydir** and all its subdirectories are saved on a tape in the default tape device:

```
$ tar c mydir
```

In this example, the **mydir** directory and all its files and subdirectories are extracted from the default tape device and placed in the user's working directory:

```
$ tar x mydir
```

NOTE *There are other archive programs you can use such as cpio, pax, and shar. However, tar is the one most commonly used for archiving application software.*

File Compression: gzip, bzip2, and zip

Several reasons exist for reducing the size of a file. The two most common are to save space or, if you are transferring the file across a network, to save transmission time. You can effectively reduce a file size by creating a compressed copy of it. Anytime you need the file again, you decompress it. Compression is used in combination with archiving to enable you to compress whole directories and their files at once. Decompression generates a copy of the archive file, which can then be extracted, generating a copy of those files and directories. File Roller provides a GUI for these tasks.

Compression with gzip

Several compression utilities are available for use on Linux and Unix systems. Most software for Linux systems uses the GNU gzip and gunzip utilities. The gzip utility compresses files, and gunzip decompresses them. To compress a file, enter the command **gzip** and the filename. This replaces the file with a compressed version of it with the extension **.gz**.

```
$ gzip mydata
$ ls
mydata.gz
```

To decompress a gzip file, use either **gzip** with the **-d** option or the command **gunzip**. These commands decompress a compressed file with the **.gz** extension and replace it with a decompressed version with the same root name but without the **.gz** extension. You needn't even type in the **.gz** extension; **gunzip** and **gzip -d** assume it. Table 6-7 lists the different gzip options.

```
$ gunzip mydata.gz
$ ls
mydata
```

TIP *On your desktop, you can extract the contents of an archive by locating it with the file manager and double-clicking it. You can also right-click and choose Open with Archive Manager. This will start the File Roller application, which will open the archive, listing its contents. You can then choose to extract the archive. File Roller will use the appropriate tools to decompress the archive (bzip2, zip, or gzip) if compressed, and then extract the archive (tar).*

You can also compress archived tar files. This results in files with the extensions **.tar.gz**. Compressed archived files are often used for transmitting extremely large files across networks.

```
$ gzip myarch.tar
$ ls
myarch.tar.gz
```

Option	Execution
-c	Sends compressed version of file to standard output; each file listed is separately compressed: **gzip -c mydata preface > myfiles.gz**
-d	Decompresses a compressed file; or you can use gunzip: **gzip -d myfiles.gz** **gunzip myfiles.gz**
-h	Displays help listing.
-l *file-list*	Displays compressed and uncompressed size of each file listed: **gzip -l myfiles.gz**
-r *directory-name*	Recursively searches for specified directories and compresses all the files in them; the search begins from the current working directory. When used with **gunzip**, compressed files of a specified directory are uncompressed.
-v *file-list*	For each compressed or decompressed file, displays its name and the percentage of its reduction in size.
-*num*	Determines the speed and size of the compression; the range is from –1 to –9. A lower number gives greater speed but less compression, resulting in a larger file that compresses and decompresses quickly. Thus –1 gives the quickest compression but with the largest size; –9 results in a very small file that takes longer to compress and decompress. The default is –6.

TABLE 6-7 The **gzip** Options

You can compress tar file members individually using the **tar z** option that invokes gzip. With the **z** option, tar invokes gzip to compress a file before placing it in an archive. Archives with members compressed with the **z** option, however, cannot be updated, nor is it possible to add to them. All members must be compressed, and all must be added at the same time.

The compress and uncompress Commands

You can also use the **compress** and **uncompress** commands to create compressed files. They generate a file that has a **.Z** extension and use a different compression format from gzip. The **compress** and **uncompress** commands are not that widely used, but you may run across **.Z** files occasionally. You can use the **uncompress** command to decompress a **.Z** file. The gzip utility is the standard GNU compression utility and should be used instead of **compress**.

Compressing with bzip2

Another popular compression utility is **bzip2**. It compresses files using the Burrows-Wheeler block-sorting text compression algorithm and Huffman coding. The command line options are similar to gzip by design, but they are not exactly the same. (See the bzip2 Man page for a complete listing.) You compress files using the **bzip2** command and decompress with **bunzip2**. The **bzip2** command creates files with the extension **.bz2**. You can use **bzcat** to output compressed data to the standard output. The **bzip2** command compresses files in blocks and enables you to specify their size (larger blocks give you greater compression). As when using gzip, you can use bzip2 to compress tar archive files. The following example compresses the **mydata** file into a bzip compressed file with the extension **.bz2**:

```
$ bzip2 mydata
$ ls
mydata.bz2
```

To decompress, use the **bunzip2** command on a bzip file:

```
$ bunzip2 mydata.bz2
```

Using Zip

Zip is a compression and archive utility modeled on PKZIP, which was used originally on DOS systems. Zip is a cross-platform utility used on Windows, Mac, MS-DOS, OS/2, Unix, and Linux systems. Zip commands can work with archives created by PKZIP and can use Zip archives. You compress a file using the **zip** command. This creates a Zip file with the **.zip** extension. If no files are listed, **zip** outputs the compressed data to the standard output. You can also use the **-** argument to have **zip** read from the standard input. To compress a directory, you include the **-r** option. The first example archives and compresses a file:

```
$ zip mydata
$ ls
mydata.zip
```

The next example archives and compresses the **reports** directory:

```
$ zip -r reports
```

A full set of archive operations is supported. With the **-f** option, you can update a particular file in the Zip archive with a newer version. The **-u** option replaces or adds files, and the **-d** option deletes files from the Zip archive. Options also exist for encrypting files, making DOS-to-Unix end-of-line translations and including hidden files.

To decompress and extract the Zip file, you use the **unzip** command.

```
$ unzip mydata.zip
```

This page intentionally left blank

PART III

Desktop

CHAPTER 7
The X Window System, Xorg, and Display Managers
CHAPTER 8
GNOME
CHAPTER 9
KDE

This page intentionally left blank

CHAPTER 7

The X Window System, Xorg, and Display Managers

Linux and Unix systems use the same standard underlying graphics utility known as the X Window System, also known as X or X11. This means that, in most cases, an X-based program can run on any of the window managers and desktops. X-based software is often found at Linux or Unix FTP sites in directories labeled **X11**. You can download these packages and run them on any window manager running on your Linux system. Some may already be in the form of Linux binaries that you can download, install, and run directly. Netscape is an example. Others are in the form of source code that can easily be configured, compiled, and installed on your system with a few simple commands. Some applications, such as Motif applications, may require special libraries.

The X Window System is designed for flexibility—you can configure it in various ways. You can run the X Window System on almost all the video cards currently available. The X Window System is not tied to any specific desktop interface. It provides an underlying set of graphical operations that user interface applications such as window managers, file managers, and even desktops can use. A window manager uses these operations to construct widgets for manipulating windows, such as scroll bars, resize boxes, and close boxes. Different window managers can construct them to appear differently, providing interfaces with different appearances. All window managers work on the X Window System. You can choose from a variety of different window managers, and each user on your system can run a different window manager, each using the same underlying X Window System graphics operations. You can even run X programs without any window or file managers.

To run the X Window System, you need to install an X Window System server. Free versions of X Window System server software are provided by both the original XFree86 project (**xfree86.org**) and the later X.org Foundation (**www.x.org**). The XFree86 project, though open source and free, uses its own license. For this reason the X.org project branched off from it to develop an entirely GNUs Not Unix (GNU) public-licensed version of the X Window System. Currently, the X.org version is used on most Linux distributions. The configuration for both implementations remains the same, with just the name of the configuration file having changed from **Xfree86.conf** to **xorg.conf** for the X.org Foundation's version. The two groups also use different naming conventions for their releases. XFree86

uses a its own numbering, currently 4.6, whereas X.org conforms to the X Window System releases, currently X11R7.2. This chapter focuses on the X.org version, as it is the more widely used one. Keep in mind that the configuration and organization are much the same for XFree86.

Once you install the Xorg server, you must provide configuration information about your monitor, mouse, and keyboard. This information is then used in a configuration file called **/etc/X11/xorg.conf**, which includes technical information best generated by an X Window System configuration program, such as Xorgconfig, xlizard, or XF86Setup. When you configured the X Window System when you installed your system, this file was automatically generated.

You can also configure your own X interface using the **.xinitrc** and **/etc/X11/xinit/xinitrc** configuration files, where window managers, file managers, and initial X applications can be selected and started. And you can use a set of specialized X commands to configure your root window, load fonts, or configure X Window System resources, such as setting the color of window borders. You can also download X utilities from online sources that serve as Linux mirror sites, usually in their **/pub/Linux/X11** directory. If you have to compile an X application, you may have to use special procedures, as well as install support packages. An official source for X Window System news, tools, and window managers is **www.x.org**. Here you can find detailed information about X Window System features, along with compliant desktops and window managers.

The X Window System was developed and is maintained by The Open Group (TOG), a consortium of over a hundred companies, including Sun, HP, IBM, Motorola, and Intel (**opengroup.org**). Development is currently managed by the X.org group on behalf of the TOG. X.org is a nonprofit organization that maintains the existing X Window System code. X.org periodically provides free official Window System update releases to the general public. It controls the development of the X11R6 specifications, working with appropriate groups to revise and release updates to the standard, as required. Xorg is a freely distributed version of X Window System servers used on most Linux systems. You can find out more about Xorg at **www.x.org**.

The X Protocol

The X protocol was developed for Unix systems in the mid-1980s to provide a network-transparent graphical user interface (GUI). The X protocol organizes display operations into a client and server relationship, in which a client submits display requests to a server. The client is known as an X client and the server as an X server. The client, in this case, is an application, and the server is a display. This relationship separates an application from the server. The application acts as a client sending requests to the server, which then does the actual work of performing the requested display operation. This has the advantage of letting the server interact with the operating system and its devices, whereas the application need know nothing of these details. An application operating as an X client can display on any system that uses an X server. In fact, a remote X client can send requests to have an X server on a local machine perform certain display operations. In effect, the X server/client relationship is inverted from the way we normally think of servers. Usually, several client systems access a single server. In the X server model, you have each system operating as an X server that can access a single system that holds X client programs.

Xorg

The X.org Foundation (**www.x.org**) is a nonprofit organization that provides free X Window System servers and supporting materials for several operating systems on PCs and other microcomputers. The X server, client programs, and documentation supplied by the X.org Foundation are commonly referred to as Xorg. The Xorg server is available free and includes source code. The project is funded entirely by donations.

Xorg uses one server, called the Xorg X server, with additional driver packages for your specific video card. You need to install only the Xorg X server package along with basic support packages such as those for fonts, as well as the driver for your particular video card, like the xf86-video-ati-X11R6 for ATI cards or xf86-video-nv-X11R6 for Nvidia cards. The Xorg X server will have support for given video cards and monitors implemented as static libraries or as modules it can load as needed. Currently, the Xorg X server supports the Intel, Alpha, PowerPC, and Sparc platforms. The Xorg server supports a wide range of video cards and monitors, including monochrome, VGA, and Super VGA, and accelerated video cards.

Your Linux distribution will normally notify you of any updates for Xorg through their update tools. Updates can then be automatically downloaded and installed. It's always preferable to download from your Linux distribution sites, since those packages may be modified to work better with your system. The entire Xorg software release includes the Xorg X server and its modules, along with several supporting packages such as those for fonts and configuration files. Table 7-1 lists the current Xorg packages. Alternatively, you can download the source code for new releases at the X.org website. For the source code versions, it is strongly recommended that you use the Xinstall.sh installer. Xinstall.sh will query for installation information and then download and install all needed Xorg components.

In addition to the server, Xorg includes support programs and development libraries. Xorg applications and servers are installed in the **/usr/bin** directory. Supporting libraries, such as the specific video card module needed, are installed in the **/usr/X11R6/lib** directory. Documentation for different packages can be found at **/usr/share/doc**, with package directories beginning with the prefix **xorg**. A detailed hard copy documentation of all X.org components can be found at **/usr/share/X11/doc**. The Man page for the X.org server is **Xorg**, and the server application is **/usr/bin/Xorg**. Configuration files are placed in the **/etc/X11** directory. Applications written to support X are usually install in the **/usr/bin** directory.

Directory	Description
/usr/X11R6/lib	Supporting libraries
/usr/bin	Programs (X Window System clients and servers)
/usr/include/X11	Development header files
/usr/share/man/X11	Man pages
/usr/share/X11/doc	Documentation
/usr/share/X11	System X11 configuration and support files
/etc/X11	Configuration files

TABLE 7-1 Xorg Directories

Tool	Description
xorgcfg	Xorg screen-based X Window System configuration tool
Xorg -configure	Xorg X Window System configuration tool that is built into the Xorg X server
xorgconfig	Older Xorg configuration tool
Sax2	SUSE X Window System configuration tool
/etc/X11/xorg.conf	The X Window System configuration file; edited by the configuration tools

TABLE 7-2 X Window System Configuration Tools

You can also find the **Xorg** server and support programs there. Table 7-2 lists Xorg configuration directories.

NOTE *Xorg now includes Direct Rendering Interface (DRI) and OPenGL support (GLX) for 3-D cards such as ATI and Nvidia.*

You can use X servers to run X Window System applications on a remote system. When you access a remote system, you can have the X server on that system generate a new display for you to run the remote X application. Every X server has a display name consisting of a hostname, a display number, and a screen number. These are used by an application to determine how to connect to the server and the screen it should use.

```
hostname:displaynumber.screennumber
```

The hostname is the host where the X server is physically located. The display number is the number of the display being managed by the X server. On a local workstation, there is usually only one display. However, on a multiuser system where several terminals (each with its own keyboard and mouse) are connected to a single system, each terminal is its own display with its own display number. This way, several users can be running X applications at the same time off the same X server. If your system has two or more monitors sharing the same keyboard and mouse, a different screen number will be applied to each monitor, though they will have the same display number.

The display a user is currently using is listed as the DISPLAY environment variable. On a single-user system, you will find that the display entry begins with a colon and is followed by a 0, as shown here. This indicates that the X server is on the local system (not a remote host) and has the display number of 0.

```
$ echo $DISPLAY
:0
```

To use a remote X application, you have to change the display name for the DISPLAY variable. You can do this manually by assigning a new hostname and display number to the variable, or you can use the **xon** script:

```
$ DISPLAY=rabbit.mytrek.com:0
$ export DISPLAY
```

You can also use the **-display** option when invoking an X application to specify the remote X server to use:

```
$ xterm -display rabbit.mytrek.com:0
```

Xorg Configuration: /etc/X11/xorg.conf

The Xorg servers provide a wide range of hardware support, but it can be challenging to configure. You can consult the X Window HOWTO documents at **tldp.org** or in the **/usr/share/doc/** directory for most distributions. There are also Man pages for Xorg and xorg.conf, and documentation and FAQs are available at **www.x.org**. The configuration file used for your Xorg server is called **xorg.conf**, located in the **/etc/X11** directory. **xorg.conf** contains all the specifications for your graphics card, monitor, keyboard, and mouse. To configure the **xorg.conf** file, you need specific information on hand about your hardware. For your monitor, you must know the horizontal and vertical sync frequency ranges and bandwidth. For your graphics card, you have to know the chipset, and you may even need to know the clocks. For your mouse, you should know whether it is Microsoft-compatible or some other brand, such as Logitech. Also, know the port to which your mouse is connected.

Although you can create and edit the file directly, it is preferable to use your distribution's display configuration tool. Xorg will now automatically detect your setup and generate an appropriate xorg.conf file. It can even start up without an **xorg.conf** configuration file. Table 7-2 lists these various configuration tools and files.

Alternatively, you can use an Xorg configuration utility built into the Xorg server. You use the **Xorg** command with the **-configure** option. This will automatically detect and generate an **xorg.conf** configuration file. The file will be named **xorg.conf.new** and placed in your root directory. To use this command you first have to exit the X server. This involves changing runlevels, changing from a graphical interface to the command line interface. On many distributions the command line interface runs on runlevel 3 (exceptions are Debian and Ubuntu). You can use the **telinit** command to change runlevels. If the command line interface uses runlevel 3, for example, you use the **telinit 3** command to change to it. This exits the desktop and prompts you to login using the command line interface.

```
telinit 3
```

Login as root and then run the **Xorg -configure** command.

```
Xorg -configure
```

You can then test out the new **xorg** configuration file with X to see if it works. Use the **-config** option. Once it does work, you can rename the original and then rename the new one as **/etc/X11/xorg.conf**.

```
X -config /root/xorg.conf.new
```

Alternatively, you can use the xorgcfg or the older xorgconfig. With these tools, you simply answer questions about your hardware or select options, and the program generates the appropriate **/etc/X11/xorg.conf** file. For a difficult configuration, you will have to edit the **xorg.conf** file directly. Usually only a few small edits to the automatically generated file are needed.

The **/etc/X11/xorg.conf** file is organized into several parts. You can find a detailed discussion of all these sections and their entries in the **xorg.conf** Man page. All of these are set by the XF86Setup program. For example, the Monitor screen generates the Monitor section in the **xorg.conf** file, the Mouse screen generates the Input Device section for the mouse, and so on. A section in the file begins with the keyword **Section,** followed by the name of the section in quotes. The section ends with the term **EndSection**. Comments have a # sign at the beginning of the line. The different kinds of sections are listed here.

Section	Description
Files	Directories for font and rgb files
Module	Dynamic module loading
ServerFlags	Miscellaneous options
Input Device	Mouse and keyboard configuration
Monitor	Monitor configuration (set horizontal and vertical frequencies)
Device	Video card configuration
Screen	Configure display, setting virtual screen, display colors, screen size, and other features
ServerLayout	Specify layout of screens and input devices

Entries for each section begin with a data specification, followed by a list of values. With release 4.0, many former data specifications are implemented using the Option entry. You enter the keyword **Option,** followed by the data specification and its value. For example, the keyboard layout specification, XkbLayout, is now implemented using an **Option** entry as shown here:

```
Option "XkbLayout" "us"
```

Although you can directly edit the file using a standard text editor, relying on the setup programs such as **xorgcfg** to make changes is always best. You won't ever have to touch most of the sections, but in some cases, you'll want to make changes to the Screen section located at the end of the file. To do so, you edit the file and add or change entries in the Screen section. In the Screen section, you can configure your virtual screen display and set the number of colors supported. Because the Screen section is the one you would most likely change, it is discussed first, even though it comes last, at the end of the file.

Screen

A Screen section begins with an Identifier entry to give a name to this Screen. After the Identifier entry, the Device and Monitor entries specify the monitor and video card you are using. The name given in the Identifier entry in these sections is used to reference those components.

```
Section "Screen"
    Identifier "Screen0"
    Device "Videocard0"
    Monitor "Monitor0"
    DefaultDepth 24
    Subsection "Display"
```

```
        Viewport   0 0
        Depth 32
        Modes "1024x768" "1920x1200"
    EndSubSection
EndSection
```

The Screen section has Display subsections, one for each depth supported. Whereas the previous sections configured hardware, the Display subsection configures display features, such as the number of colors displayed and the virtual screen size. Two main entries exist: Depth and Modes. The Depth entry is the screen resolution: 8, 16, and 24. You can add the DefaultDepth entry to set the default color depth to whatever your X server supports: 8 for 256 K, 16 for 32 K, and 24 for 16 M. Modes are the modes allowed given the resolution. You can also add to the Virtual entry to specify the size of the virtual screen. You can have a virtual screen larger than your display area. When you move your mouse to the edge of the displayed screen, it scrolls to that hidden part of the screen. This way, you can have a working screen much larger than the physical size of your monitor. The physical screen size for a 17-inch monitor is usually 1024 × 768. You can set it to 1152 × 864, a 21-inch monitor size, with a Virtual entry.

Any of these features in this section can be safely changed. In fact, to change the virtual screen size, you must modify this section. Other sections in the **xorg.conf** file should be left alone, unless you are certain of what you are doing.

Normally, these entries are automatically detected. However, on some monitor and video card combinations, the screen resolution could be incorrectly detected, leaving you with only lower resolutions. To fix this you may have to place a Modes entry in the Screen Display section, listing your possible resolutions. Automatically generated versions of **xorg.conf** will not have a Modes entry. Alternatively, you can install a vendor-provided version of your X11 driver if available, such as those from Nvidia or ATI.

```
    Modes "1024x768" "1980x1200"
```

Files, Modules, and ServerFlags

The Files, Modules, and ServerFlags are usually not needed for a simple automatic configuration. More complex configuration may need them. **`Xorg -configure`** will generate system entries for them.

The configuration section lists different directories for resources that Xorg needs. For example, to specify the location where RGB color data is listed, a line begins with the data specification **RgbPath,** followed by the pathname for that **rgb** color data file. Fonts for the X Window System are handled by the XFS server whose configuration files is located in the **X11/fs** directory. Alternatively, specific fonts can be listed in the Files section using the **FontPath** option **ModulePath** entry specifies the pathname for the modules directory. This directory will hold the modules for specific video card drivers. A sample of these entries is shown here:

RgbPath "/usr/share/X11/rgb"
ModulePath "/usr/lib/xorg/modules"

A specific X Window System font can be desingated with a FontPath entry. Here is a sample of one such entry, using a font located in the /usr/share/fonts/X11 directory (Ubuntu).

```
FontPath   "/usr/share/fonts/X11/75dpi"
```

If no FontPaths are specified and the XFS server is not used, the X server falls back on default font paths already compiled into the X server (see the **xorg.conf** Man page for more details). The Module section specifies modules to be dynamically loaded, and the Load entry loads a module, which is used to load server extension modules and font modules. This is a feature introduced with version 4.0 that allows X server components that extend the functionality of the X server to be loaded as modules. This feature provides for easy updating, letting you upgrade modules without having to replace the entire X server. For example, the extmod module contains miscellaneous extensions to enable commonly used functions in the X server. In the following example, the extmod module is loaded that contains a set of needed extensions. Of special note are the dri, glx, and GLcore modules. These provide accelerated support for 3-D cards. See the **xorg.conf** Man page for more details.

```
Load "extmod"
Load "dri"
Load "glx"
Load "GLcore"
```

Several flags can be set for the Xorg server. These are now implemented as options. (You can find a complete listing in the **xorg.conf** Man page.) For example, the **`BlankTime`** value specifies the inactivity timeout for the screen saver. **`DontZap`** disables the use of CTRL-ALT-BACKSPACE to shut down the server. **`DontZoom`** disables switching between graphics modes. You create an Option entry with the flag as the option. The following example sets the server flag for the screen saver inactivity timeout:

```
Option "BlankTime " "30"
```

Input Device

With version 4.0, the Input Device section replaced the previous Keyboard, Pointer, and XInput sections. To provide support for an input device such as a keyboard, you create an Input Device section for it and enter Identifier and Driver entries for the device. For example, the following entry creates an Input Device section for the keyboard:

```
Section "Input Device"
 Identifier "Keyboard 0"
 Driver "kbd"
```

Any features are added as options, such as keyboard layout or model. A large number of options exist for this section. Consult the **xorg.conf** Man pages for a complete listing. The following example shows an entire keyboard entry with autorepeat, keyboard model (XkbModel), and keyboard layout (**`XkbLayout`**) options entered:

```
Section "InputDevice"
 Identifier "Keyboard 0"
 Driver "kbd"
 Option "AutoRepeat" "500 5"
 Option "XkbModel" "pc105"
 Option "XkbLayout" "us"
EndSection
```

You create an Input Device section for your mouse and any other pointer devices. However, on systems that use desktops like GNOME and KDE, mouse configuration is handled directly by the desktop. There is no Xorg configuration.

The mouse section has only a few entries, with some tailored for specific types of mice. Features are defined using Option entries. The Protocol option specifies the protocol your mouse uses, such as PS/2, Microsoft, or Logitech. The Device option is the pathname for the mouse device. The following example shows a standard Pointer section for a three-button PS/2 mouse. The device file is **/dev/mouse**.

```
Section "InputDevice"
 Identifier "Mouse 1"
 Driver "mouse"
 Option "Protocol" "PS/2"
 Option "Device" "/dev/mouse"
 Option "Emulate3Buttons" "off"
EndSection
```

Monitor

A Monitor section should exist for each monitor used on your system. The vertical and horizontal frequencies must be accurate or you can damage your monitor. A Monitor section begins with entries that identify the monitor, such as vendor and model names. The HorizSync and VerRefresh entries are where the vertical and horizontal frequencies are specified. Most monitors can support a variety of resolutions. Those resolutions are specified in the Monitor section by ModeLine entries. A ModeLine entry exists for each resolution. The ModeLine entry has five values, the name of the resolution, its dot clock value, and then two sets of four values, one for the horizontal timing and one for the vertical timing, ending with flags. The flags specify different characteristics of the mode, such as Interlace, to indicate the mode is interlaced, and +hsync and +vsync to select the polarity of the signal.

```
ModeLine "name" dotclock horizontal-freq vertical-freq flags
```

A sample of a ModeLine entry is shown here. Leaving the entire Monitor section alone is best; rely, instead, on the entries generated by XF86Setup.

```
Modeline "800x600" 50.00 800 856 976 1040 600 637 643 666 +hsync +vsync
```

Commonly used entries for the Monitor section are listed here:

Option	Description
Identifier	A name to identify the monitor
VendorName	Manufacturer
ModelName	The make and model
HorizSync	The horizontal refresh frequency; can be a range or series of values
VerRefresh	Vertical refresh frequency; can be a range or series of values
Gamma	Gamma correction
ModeLine	Specifies a resolution with dot clock, horizontal timing, and vertical timing for that resolution

A sample Monitor section is shown here:

```
Section "Monitor"
   Identifier "Monitor0"
   VendorName "Dell 2405FPW (Analog)"
   ModelName "Unknown"
   HorizSync 30 - 83.0
   VertRefresh 56 - 76.0
   Option "dpms"
EndSection
```

Device

The Device section specifies your video card. It begins with an Identifier entry and an entry for the video card driver. The following example creates an Identifier for an Nvidia card called "Videocard0" and then specifies that the nv driver (Nvidia) is to be used for it:

```
Identifier " Videocard0"
Driver "nv"
```

Further entries identify the card, such as VendorName, BoardName, and Chipset. The amount of video RAM is indicated in the VideoRam entry. The Clocks entry lists your clock values. Many different entries can be made in this section, such as Ramdac for a Ramdac chip, if the board has one, and MemBase for the base address of a frame buffer, if it is accessible. See the **xorg.conf** Man pages for a detailed list and descriptions.

Although you can safely change a VideoRam entry—for example, if you add more memory to your card—changing the Clocks entry is not safe. If you get the clock values wrong, you can easily destroy your monitor. Rely on the clock values generated by xorgcfg or other Xorg setup programs. If the clock values are missing, it means that the server will automatically determine them. This may be the case for newer cards. A sample Device entry is shown here:

```
Section "Device"
 Identifier "Videocard0"
 Driver "nv"
EndSection
```

Depending on the level of detection, more detailed information may be generated:

```
Identifier  "Card0"
Driver      "nouveau"
VendorName  "nVidia Corporation"
BoardName   "NV43 [GeForce 6600]"
BusID       "PCI:3:0:0"
```

ServerLayout

A ServerLayout section lets you specify the layout of the screens and the selection of input devices. The ServerLayout sections may also include options that are normally found in the ServerFlags section. You can set up several ServerLayout sections and select them from the command line. The following example shows a simple ServerLayout section for a basic configuration:

```
Section "ServerLayout"
 Identifier "single head configuration"
 Screen 0 "Screen0" 0 0
 InputDevice "Keyboard0" "CoreKeyboard"
EndSection
```

Multiple Monitors

If you have more than one video card with a monitor connect to each, then your X server will detect and implement them, each with their own device and monitor entries. Monitors connected to the same video card require more complex configuration. In effect, you have two monitors using the same device, the same video card. With the Nvidia and ATI proprietary drivers, you can use their configuration tools to configure separate Monitors. These drivers also support extended desktops, where one monitor can display an extension of another (TwinView on Nvidia).

Standard X servers for most cards also support multiple displays. Most distributions provide display configuration tools to let you easily configure separate displays. You can also implement an extended desktop using the X server Xinerama service.

To configure two separate monitors on a single Nvidia card, corresponding Device, Screen, and Monitor sections are set up for each monitor, with the Screen sections connecting the Monitor and Device sections. The ServerLayout section lists both screens. For an extended desktop, the TwinView option is set in the Device section, with specifications for each monitor. Check the Nvidia readme for Linux installation for more details.

ATI cards follow much the same format. The MergedFB option implements the extended desktop option of the ATI X driver, and the DesktopSetup option is used for the ATI proprietary driver.

X Window System Command Line Arguments

You can start up any X Window System application within either an **.xinitrc** or **.xsession** script or on the command line in an Xterm window. Some distributions, including Mandrake and Red Hat, allow users to place X Window System startup applications in an **.Xclients** file that is read by the **.xinitrc** script. Most X Window System applications take a set of standard X Window System arguments used to configure the window and display the application uses. You can set the color of the window bars, give the window a specific title, and specify the color and font for text, as well as position the window at a specific location on the screen. Table 7-3 lists these X Window System arguments. They are discussed in more detail in the X Man pages, **man X**.

One commonly used argument is **`-geometry`**. This takes an additional argument that specifies the location on the screen where you want an application's window displayed. In the next example, the xclock X Window System application is called with a **`-geometry`** argument. A set of up to four numbers specifies the position. The value +0+0 references the upper-left corner. There, you see the clock displayed when you start up the X Window System. The value -0-0 references the upper-right corner.

```
& xclock -geometry +0+0 &
```

X Window Application Configuration Arguments	Description
-bw *num*	Border width of pixels in frame
-bd *color*	Border color
-fg *color*	Foreground color (for text or graphics)
-bg *color*	Background color
-display *display-name*	Displays client to run on; displays name consisting of hostname, display number, and screen number (see X Man pages)
-fn *font*	Font to use for text display
-geometry *offsets*	Location on screen where X Window System application window is placed; offsets are measured relative to screen display
-iconic	Starts application with icon, not with open window
-rv	Switches background and foreground colors
-title *string*	Title for the window's title bar
-name *string*	Name for the application
-xrm *resource-string*	Specifies resource value

TABLE 7-3 Configuration Options for X Window System–Based Applications

With the **-title** option, you can set the title displayed on the application window. Notice the use of quotes for titles with more than one word. You set the font with the **-fn** argument, and the text and graphics color with the **-fg** argument. **-bg** sets the background color. The following example starts up an Xterm window with the title "My New Window" in the title bar. The text and graphics color is green, and the background color is gray. The font is Helvetica.

```
$ xterm -title "My New Window" -fg green -bg gray -fn /usr/fonts/helvetica &
```

X Window System Commands and Configuration Files

The X Window System uses several configuration files, as well as X commands to configure your X Window System. Some of the configuration files belong to the system and should not be modified. Each user can have their own set of configuration files, however, such as **.xinitrc**, **.xsession**, and **.Xresources**, that can be used to configure a personalized X Window System interface. The **fs** directory holds the configuration file for X Window System fonts. An **.Xclients** file can hold X Window System startup applications. These configuration files are automatically read and executed when the X Window System is started up with either the **startx** command or an X display manager, such as XDM or GDM. Within these configuration files, you can execute X commands used to configure your system. With commands such as **xset** and **xsetroot**, you can add fonts or control the display of your root window. Later in this chapter, Table 7-4 provides a list of X Window System

X Window Commands	Explanations
xterm	Opens a new terminal window
xset	Sets X Window System options; see Man pages for complete listing **-b** Configures bell **-c** Configures key click **+fp** *fontlist* Adds fonts **-fp** *fontlist* Removes fonts **led** Turns on or off keyboard LEDs **m** Configures mouse **p** Sets pixel color values **s** Sets the screen saver **q** Lists current settings
xsetroot	Configures the root window **-cursor** *cursorfile maskfile* Sets pointer to bitmap pictures when pointer is outside any window **-bitmap** *filename* Sets root window pattern to bitmap **-gray** Sets background to gray **-fg** *color* Sets color of foreground bitmap **-bg** *color* Sets color of background bitmap **-solid** *color* Sets background color **-name** *string* Sets name of root window to *string*
xmodmap	Configures input devices; reads the **.Xmodmap** file **-pk** Displays current keymap **-e** *expression* Sets key binding keycode NUMBER = KEYSYMNAME Sets key to specified key symbol keysym KEYSYMNAME = KEYSYMNAME Sets key to operate the same as specified key pointer = NUMBER Sets mouse button codes
xrdb	Configures X Window System resources; reads the **.Xresources** file
xdm	X Window System display manager; runs the Xorg server for your system; usually called by **xinitrc**
startx	Starts X Window System by executing **xinit** and instructing it to read the **xinitrc** file
xfs *config-file*	The X Window System font server
mkfontdir *font-directory*	Indexes new fonts, making them accessible by the font server
xlsfonts	Lists fonts on your system
xfontsel	Displays installed fonts
xdpyinfo	Lists detailed information about your X Window System configuration
xinit	Starts X Window System, first reading the system's **xinitrc** file; when invoked from **startx**, it also reads the user's **.Xclients** file; **xinit** is not called directly, but through **startx**
xmkmf	Creates a makefile for an X Window System application using the application's **Imakefile**; invokes **imake** to generate the makefile (never invoke **imake** directly)
xauth	Reads **.Xauthority** file to set access control to a user account through XDM from remote systems

TABLE 7-4 X Window System Commands

configuration files and commands. You can obtain a complete description of your current X configuration using the **xdypinfo** command. The X Man pages, provide a detailed introduction to the X commands and configuration files.

XFS Fonts

Most fonts are now handled directly by desktops (GNOME or KDE, **fonts:/**) using **fontconfig**, and are very easy to install. These fonts are stored in either the user's **.fonts** directory or **/usr/share/fonts**. In addition, there is also separate font support for just your X Window System, using fonts in **/usr/share/X11/fonts**. X Window System fonts are manged by the XFS font server, configured with the **/etc/X11/fs/config** configuration file. This file lists fonts in the **catalogue** entry. The X Man pages provide a detailed discussion on fonts. Fonts can be manually loaded with the **xfs** command. Before you can access newly installed fonts, you must first index them with the **mkfontdir** command. To have the fonts automatically loaded, add the directory with the full pathname to the **catalogue** entry in the XFS configuration file.

NOTE *Some recent distributions are dropping the XFS font server in place of a few specific fonts installed in a designated X11 directory.*

X Resources

Several X commands, such as **xrdb** and **xmodmap**, configure your X Window System interface. X Window System graphics configurations are listed in a resource file called **.Xresources**. Each user can have a customized **.Xresources** file in their home directory, configuring the X Window System to particular specifications. The **.Xresources** file contains entries for configuring specific programs, such as the color of certain widgets. A systemwide version called **/etc/X11/Xresources** also exists. The **.Xdefaults** file is a default configuration loaded by all programs, which contains the same kind of entries for configuring resources as **.Xresources**. An **.Xdefaults** file is accessible by programs on your system but not by those running on other systems. The **/usr/share/X11/app-defaults** directory holds files that contain default resource configurations for particular X applications, such as **Xterm**, **Xclock**, and **Xmixer**. The **Xterm** file holds resource entries specifying how an Xterm window is displayed. You can override any of these defaults with alternative entries in an **.Xresources** file in your home directory. You can create an **.Xresources** file of your own in your home directory and add resource entries to it. You can also copy the **/etc/X11/Xresources** file and edit the entries there or add new ones of your own.

Configuration is carried out by the **xrdb** command, which reads both the system's **.Xresources** file and any .**Xresources** or .**Xdefaults** file in your home directory. The **xrdb** command is currently executed in the **/etc/X11/xinit/xinitrc** script and the **/etc/X11/xdm/Xsession** script. If you create your own **.xinitrc** script in your home directory, be sure it executes the **xrdb** command with at least your own **.Xresources** file or the **/etc/X11/Xresources** file (preferably both). You can ensure this by simply using a copy of the system's **xinitrc** script as your own **.xinitrc** file, and then modifying that copy as you want. See the Man pages on **xrdb** for more details on resources. Also, you can find a more detailed discussion of Xresources, as well as other X commands, in the Man pages for X.

An entry in the **.Xresources** file consists of a value assigned to a resource, a class, or a group of resources for an application. Usually, resources are used for widgets or classes of widgets in an application. The resource designation typically consists of three elements separated by periods: the application, an object in the application, and the resource. The entire designation is terminated by a colon, and the value follows. For example, suppose you want to change the color of the hour hand to blue in the oclock application. The application is oclock, the object is clock, and the resource is hour: oclock.clock.hour. This entry looks like this:

```
oclock.clock.hour: blue
```

The object element is actually a list of objects denoting the hierarchy leading to a particular object. In the oclock example, only one object exists, but in many applications, the object hierarchy can be complex. This requires a lengthy set of objects listed to specify the one you want. To avoid this complexity, you can use the asterisk notation to reference the object you want directly, using an asterisk in place of the period. You only need to know the name of the resource you want to change. The following example sets the oclock minute and hour hands to green:

```
oclock*hour: green
oclock*minute: green
```


You can also use the asterisk to apply a value to whole classes of objects. Many individual resources are grouped into classes. You can reference all the resources in a class by their class name. Class names begin with an uppercase character. In the Xterm application, for example, the background and pointer color resources are both part of the Background class. The reference **`XTerm*Background`** changes all these resources in an Xterm window. However, specific references always override the more general ones.

You can also use the asterisk to change the values of a resource in objects for all your applications. In this case, you place an asterisk before the resource. For example, to change the foreground color to red for all the objects in every application, you enter:

```
*foreground: red
```

If you want to change the foreground color of the scroll bars in all your applications, you use:

```
*scrollbar*foreground: blue
```

The **`showrgb`** command lists the different colors available on your system. You can use the descriptive name or a hexadecimal form. Values can also be fonts, bitmaps, and pixmaps. You could change the font displayed by certain objects in, or for, graphics applications as well as change background or border graphics. Resources vary with each application. Applications may support different kinds of objects and the resources for them. Check the Man pages and documentation for an application to learn what resources it supports and the values accepted for it. Some resources take Boolean values that can turn features on or off, while others can specify options. Some applications have a default set of resource values that is automatically placed in your system's **.Xresources** or **.Xdefaults** files.

The .**Xmodmap** file holds configurations for your input devices, such as your mouse and keyboard (for example, you can bind keys such as BACKSPACE or reverse the click operations of your right and left mouse buttons). The .**Xmodmap** file used by your display manager is in the display manager configuration directory, such as **/etc/X11/xdm**, whereas the one used by **startx** is located in **/etc/X11/xinit**. Each user can create a custom **.Xmodmap** file in their home directory to configure the system's input devices. This is helpful if users connect through their own terminals to your Linux system. The **.Xmodmap** file is read by the **xmodmap** command, which performs the configuration. The **xmodmap** command first looks for an **.Xmodmap** file in the user's home directory and uses that. If no **.Xmodmap** is in the home directory, it uses the one for your display manager or **startx** command. You see entries for the **xmodmap** command in the **/etc/X11/xinit/xinitrc** file and the display manager's **Xsession** file. If you have your own **.xinitrc** or **.xsession** script in your home directory, it should execute the **xmodmap** command with either your own **.Xmodmap** file or the system's **Xmodmap** file. See the Man pages on **xmodmap** for more details.

X Commands

Usually, an **.xinitrc** or **.xsession** script has X Window System commands, such as **xset** and **xsetroot**, used to configure different features of your X Window System session. The **xset** command sets different options, such as turning on the screen saver or setting the volume for the bell and speaker. You can also use **xset** to load fonts. See the **xset** Man pages for specific details. With the **b** option and the **on** or **off** argument, **xset** turns your speaker on or off. The following example turns on the speaker:

```
xset b on
```

You use **xset** with the **-s** option to set the screen saver. With the **on** and **off** arguments, you can turn the screen saver on or off. Two numbers entered as arguments specify the length and period in seconds. The length is the number of seconds the screen saver waits before activating, and the period is how long it waits before regenerating the pattern.

The **xsetroot** command enables you to set the features of your root window (setting the color or displaying a bitmap pattern—you can even use a cursor of your own design). Table 7-5 lists the different **xsetroot** options. See the Man pages for **xsetroot** for options and details. The following **xsetroot** command uses the **-solid** option to set the background color of the root window to blue:

```
xsetroot -solid blue
```

Table 7-4 lists common X Window System commands, and Table 7-5 lists the configuration files and directories associated with the X Window System.

Display Managers: XDM, GDM, and KDM

A display manager automatically starts the X Window System when you boot your computer, displaying a login window and a menu for selecting the window manager or desktop you want to use. Options for shutting down your system are also there. Currently, you can use three display managers. The K Display Manager (KDM) is a display manager provided with the KDE. The GNOME Display Manager (GDM) comes with the GNOME desktop. The XDM is the original display manager and is rarely used on Linux systems directly.

Configuration Files	Explanation
.Xmodmap	User's X Window System input devices configuration file
.Xresources	User's X Window System resource configuration file
.Xdefaults	User's X Window System resource configuration file
.xinitrc	User's X Window System configuration file read automatically (by **xinit**, if it exists)
.Xclients or .Xsessions	User's X Window configuration file
.Xauthority	User's access controls through XDM GUI login interface
/etc/X11/	Directory that holds X Window System release 6 configuration file and subdirectories
/etc/X11/fs	System X Window System fonts configuration directory
/etc/X11/xinit/xinitrc	System X Window System initialization file; automatically read by **xinit**
/etc/X11/xinit/Xclients	System X Window System configuration file
/etc/X11/Xresources	System X Window System resources file
/etc/X11/Xmodmap	System X Window System input devices file
/usr/share/X11/rgb.txt	X Window System colors. Each entry has four fields: the first three fields are numbers for red, green, and blue; the last field is the name given to the color.
/usr/share/X11	System-managed X Window System directory for font storage and application configuration

TABLE 7-5 X Window System Configuration Files and Directories

When a system configured to run a display manager starts up, the X Window System starts up immediately and displays a login dialog box. The dialog box prompts the user to enter a login name and a password. Once they are entered, a selected X Window System interface starts up—say, with GNOME, KDE, or some other desktop or window manager. When the user quits the window manager or desktop, the system returns to the login dialog box and remains there until another user logs in. You can shift to a command line interface with the CTRL-ALT-F1 keys and return to the display manager login dialog box with CTRL-ALT-F7. To stop the X server completely, you stop the display manager, **/etc/init.d/gdm stop.**

You can also use the display manager to control access to different hosts and users on your network. The **.Xauthority** file in each user's home directory contains authentication information for that user. A display manager like XDM supports the X Display Manager Control Protocol (XDMCP). They were originally designed for systems like workstations that are continually operating, but they are also used to start up the X Window System automatically on single-user systems when the system boots.

NOTE *Most distributions will install either KDM or GDM. Distributions that favor KDE will install KDM, whereas distributions that include both GNOME and KDE will install GDM.*

A display manager is automatically run when your system starts up at the graphical runlevel. On many distributions this runlevel is 5. Your system can run at different runlevels; for example, the standard multiuser level, a nonnetwork user level, and a system administration level. The graphical runlevel is the same as the standard multiuser level, except it automatically starts up the X Window System on connected machines and activates the display manager's login screen.

During most distribution installations, your system is configured to automatically start at graphical runlevel (number 5 on many distributions), activating the display manager. If, instead, you are starting with a standard line-mode login prompt (standard multiuser), you can manually change to the display manager by changing your runlevel to number of the graphical runlevel. To do this temporarily, you can specify your runlevel with the **telinit** administration utility. The following command changes to the standard multiuser runlevel (3 on many distributions), the command line:

```
telinit 3
```

This command will change to the graphical runlevel (graphical login), the graphical login:

```
telinit 5
```

To make a runlevel the default, you have to edit the **/etc/inittab** file.

Xsession

A display manager refers to a user's login and startup of a window manager and desktop as a session. When the user quits the desktop and logs out, the session ends. When another user logs in, a new session starts. The X Window System never shuts down; only desktop or window manager programs shut down. Session menus on the display manager login window list different kinds of sessions you can start—in other words, different kinds of window managers or desktops. For each session, the **Xsession** script is the startup script used to configure a user's X Window System display and to execute the selected desktop or window manager.

Xsession is the display manager session startup script used by GDM as well as KDM and XDM. It contains many of the X commands also used in the **xinitrc** startup script used by **startx**. Commonly executed commands for all display managers and desktops are held in the **xinitrc-common** script, which Xsession runs first. The **xinitrc-common** script executes **xmodmap** and **xrdb** commands using the **.Xmodmap** and **.Xresources** files in the **/etc/X11/xinit** directory. **Xsession** saves any errors in the user's **.xsession-errors** file in their home directory. **Xsession** will also read any shell scripts located in the **/etc/X11/xinit/xinitrc.d** directory. Currently, this holds an input script to detect the kind of language a keyboard uses, as well as scripts for any additional desktop configurations like the xdg-user-dirs service implemented by some distributions.

Xsession is usually invoked with an argument indicating the kind of environment to run, such as GNOME, KDE, or a window manager like Window Maker. The option for GNOME is **gnome** and for KDE it is **kde**.

```
Xsession gnome
```

These environments are listed in the **Xsession** script within the case statement. Here you will find entries for GNOME, KDE, and the bare-boned twm window manager. GNOME is invoked directly with the **gnome-session** command, and KDE with the **startkde** command. If **Xsession** is not invoked with a specific environment, the user's home directory is checked for a **.Xsession** or **.Xclients** script. If those scripts are missing, the system **Xclients** script is used, **/etc/X11/xinit/Xclients**. Xclients will check to see if either GNOME or KDE is installed and start the one that is. If neither is installed, it uses the old **twm** window manager.

If users want to set up their own startup files, they can copy the **Xsession** file to their home directory and name it **.xsession** and then edit it. The following example shows a simplified **Xsession** script that executes the user's **.xsession** script if it exists. The user's **.xsession** script is expected to start a window manager or desktop. The following example is taken from code in the **Xclients** script, which starts up a simple twm window manager, opening a terminal window.

```
#
# Xsession

startup=$HOME/.xsession
resources=$HOME/.Xresources

if [ -f "$startup" ]; then
           exec "$startup"
        else
           if [ -f "$resources" ]; then
                xrdb -load "$resources"
           fi

   if [ -x /usr/bin/xterm ] ; then
      /usr/bin/xterm -geometry 80x50-50+150 &
   fi
   if [ -x /usr/bin/twm ] ; then
      exec /usr/bin/twm
    fi
fi
```

NOTE *As an enhancement to either* **startx** *or a display manager, you can use the X session manager (xsm). You can use it to launch your X Window System with different sessions. A* session *is a specified group of X applications. Starting with one session might start GNOME and Mozilla, while starting with another might start KDE and KOffice. You can save your session while you are using it or when you shut down. The applications you are running become part of a saved session. When you start, xsm displays a session menu for you to choose from, listing previous sessions you saved.*

The X Display Manager (XDM)

XDM manages a collection of X displays either on the local system or remote servers. XDM's design is based on the X Consortium standard XDMCP. The XDM program manages user logins, providing authentication and starting sessions. For character-based logins, a session is the lifetime of the user shell that is started up when the user logs in from the command line interface.

For XDM and other display managers, the session is determined by the session manager. The session is usually the duration of a window manager or desktop. When the desktop or window manager terminates, so does the session.

The XDM program displays a login window with boxes for a login name and password. The user logs in, and a window manager or desktop starts up. When the user quits the window manager, the X Window System restarts automatically, displaying the login window again. Authentications to control access for particular users are kept in their **.Xauthority** file.

The XDM configuration files are located in the **/etc/X11/xdm/** directory. The main XDM configuration file is **xdm-config**. Files such as **Xresources** configure how the dialog box is displayed, and **Xsetup** enables you to specify a root-window image or other windows to display. When the user starts up a session, the **Xsession** script is run to configure the user's X Window System and execute the user's window manager or desktop. This script usually calls the **.xsession** script in the user's home directory, if there is one. It holds any specific user X commands.

If you want to start XDM from the command line interface, you can enter the command **xdm** with the **-nodaemon** option. CTRL-C then shuts down XDM:

```
xdm -nodaemon
```

Table 7-6 lists the configuration files and directories associated with XDM. **xdm-errors** will contain error messages from XDM and the scripts it runs, such as **Xsession** and **Xstartup**. Check this file if you are having any trouble with XDM.

The GNOME Display Manager

GDM manages user login and GUI sessions. GDM can service several displays and generates a process for each. The main GDM process listens for XDMCP requests from remote displays and monitors the local display sessions. GDM displays a login window with boxes for entering a login name and password and also displays entries for sessions

Filenames	Description
/etc/X11/xdm	XDM configuration directory
xdm-config	XDM configuration file
Xsession	Startup script for user session
Xresource	Resource features for XDM login window
Xsetup	Sets up the login window and XDM login screen
Xstartup	Session startup script
xdm-errors	Errors from XDM sessions
.xsession	User's session script in the home directory; usually executed by **Xsession**
Xreset	Resets the X Window System after a session ends
.Xauthority	User authorization file where XDM stores keys for clients to read

TABLE 7-6 The XDM Configuration Files and Directories

and shutdown submenus. The sessions menu displays different window managers and desktops you can start up, such as GNOME or KDE.

When the GDM starts up, it shows a login window with a box for login. Various GDM themes are available, which you can select using the GDM configuration tool. Three pop-up menus are located at the center of the screen, labeled Language, Options, and Shutdown. To log in, enter your username in the entry box labeled Username and press ENTER. You will be prompted to enter your password. Do so, and press ENTER. By default, the GNOME desktop is then started up.

When you log out from the desktop, you return to the GDM login window. To shut down your Linux system, click the Shutdown button. To restart, select Restart from the Options menu. Alternatively, you can also shut down from GNOME. From the System menu, select the Shutdown entry. GNOME will display a dialog screen with the buttons Suspend, Shutdown, or Reboot. Shutdown is the default and will occur automatically after a few seconds. Selecting Reboot will shut down and restart your system.

From the Options menu, you can select the desktop or window manager you want to start up. Here you can select KDE to start up the K Desktop, for example, instead of GNOME. On Fedora, both KDE and GNOME will use similar themes, appearing much the same. The Language menu lists a variety of different languages that Linux supports. Choose one to change the language interface.

GDM Configuration: gdmsetup

If you want to change the GDM login screen, you can use **gdmsetup**. This is often accessible from the GNOME System menu and labeled Login Screen. With **gdmsetup** you can set the background image, icons to be displayed, the theme to use, users to list, and even the welcome message. Login screens can be configured for local or remote users. You can choose between a plain screen, a plain screen with a face browser, or a themed screen. The local panel lets you select what screen to use for local logins, as well as browse among available themes. From the remote panel you can select plain or plain with browser or use the same configuration as your local logins.

On the Users panel, you can select which users you want displayed when using a face browser. On the local panel, you can choose from a number of themes. You can also opt to have the theme randomly selected.

On the security panel, you can set up an automatic login, skipping the login screen on startup. You can even set a timed login, automatically logging in a specific user after displaying the login screen for a given amount of time. In the Security segment of the panel, you can set security options, such as whether to allow root logins or allow TCP (Internet) access, as well as setting the number of allowable logins. Click the Configure X Server button on this panel to open a window for configuring X server access. Check the GNOME Display Manager Reference Manual, accessible with the Help button, for details.

GDM Configuration Files

The GDM configuration files are located in the **/etc/gdm** and **/usr/share/gdm** directories. Its uses two configuration file where various options are set, such as the logo image and welcome text to display. The **defaults.conf** in **/usr/share/gdm** should not be edited. It will be overwritten on a GDM upgrade and is ignored if a **custom.conf** file is set up. The **custom.conf** in **/etc/gdm** is where you can set up a custom configuration. Initially, the **custom.conf** file is empty, though it contains detailed comments.

The **/etc/gdm** directory contains five subdirectories: **Init**, **modules**, **Postlogin**, **PostSession**, and **PreSession**. You can easily configure GDM by placing or editing files in these different directories. The **Init** directory contains scripts that are executed when GDM starts up. This directory contains a **Default** script that holds X commands, such as setting the background. These are applied to the screen showing the GDM login window. The **modules** directory holds keyboard and mouse configurations for alternative and enhanced access, like the desktop magnifier.

The **PreSession** directory holds any presession commands to execute, while the **PostSession** directory holds scripts for commands you want executed whenever a session ends. Both have **Default** scripts. None of the **Init**, **PreSession**, or **PostSession** scripts are necessary. The **PostLogin** directory holds scripts to execute after login but before the X Window System session begins. A sample script is provided.

For GDM, the login window is generated by a program called *the greeter.* Initially, the greeter looks for icons for every user on the system, located in the **.gnome/photo** file in users' home directories. Clicking the icon automatically displays the name of the user in the login box. The user can then enter the password and click the Login button to log in.

Table 7-7 lists the configuration files and directories associated with GDM.

The K Display Manager (KDM)

The K Display Manager (KDM) also manages user logins and starts X Window System sessions. KDM is derived from the XDM, using the same configuration files. The KDM login window displays a list of user icons for users on the system. A user can click their icon and that user's name then appears in the login box. Enter the password and click Go to log in. The Session menu is a drop-down menu showing possible sessions. Click the Shutdown button to shut down the system.

You configure KDM using the KDM Configuration Manager located on the KDE root user desktop. Panels exist for configuring the background, logo, and welcome message, as well as for adding icons for users on the system. To add a new session entry in the Session menu, enter the name for the entry in the New Type box on the Sessions panel and click Add.

KDM uses the same configuration files that are located in **/etc/kde/kdm**. Many of the scripts are links to files in the XDM configuration directory, **/etc/X11/xdm**. These include links to the XDM **Xsession**, **Xresources**, and **Xsetup**, among others. KDM uses its own

Directory or Filename	Description
/etc/gdm	GDM configuration directory
/usr/share/gdm	GDM configuration directory for default settings and themes
defaults.conf	GDM default configuration file, **/usr/share/gdm**
custom.conf	GDM custom configuration file, **/etc/gdm**
Init	Startup scripts for configuring GDM display
PreSession	Scripts execute at start of session
PostSession	Scripts execute when session ends
PostLogin	Scripts execute after login

TABLE 7-7 The GDM Configuration Files and Directories

Xstartup, and the resources used to control how the KDM login window is displayed are set in the **/etc/X11/xdm//kdmrc** file (**/etc/kde/kdm/kdmrc** links to it). This is the file configured by the KDM Configuration Manager.

X Window System Command Line Startup: startx, xinit, and xinitrc

If you start Linux with the command line interface, then, once you log in, you can use the **startx** command to start the X Window System and your window manager and desktop. The **startx** command uses the **xinit** command to start the X Window System; its startup script is **/etc/X11/xinit/xinitrc**.

The X Window System can be started from the command line interface using the **xinit** command. You do not invoke the **xinit** command directly, but through the **startx** command, which you always use to start the X Window System. The **startx** command is a shell script that executes the **xinit** command. The **xinit** command, in turn, first looks for an X Window System initialization script called **.xinitrc** in the user's home directory. If no **.xinitrc** script is in the home directory, **xinit** uses **/etc/X11/xinit/xinitrc** as its initialization script. Both **.xinitrc** and **/etc/X11/xinit/xinitrc** have commands to configure your X Window System server and to execute any initial X commands, such as starting up the window manager. You can think of the **/etc/X11/xinit/xinitrc** script as a default script. In addition, many systems use a separate file named **Xclients,** where particular X applications, desktops, or window managers can be specified. These entries can be directly listed in an **xinitrc** file, but a separate file makes for a more organized format. The **Xclients** files are executed as shell scripts by the **xinitrc** file. A user version, as well as a system version, exists: **.Xclients** and **/etc/X11/init/Xclients**. The user's home directory is checked for the **.Xclients** file and, if missing, the **/etc/X11/xinit/Xclients** file is used.

Most distributions do not initially set up **.xinitrc** or **.Xclients** scripts in any of the home directories. These must be created by a particular user who wants one. Each user can create a personalized **.xinitrc** script in their home directory, configuring and starting up the X Window System as wanted. Until a user sets up an **.xinitrc** script, the **/etc/X11/xinit/xinitrc** script is used and you can examine this script to see how the X Window System starts. Certain configuration operations required for the X Window System must be in the **.xinitrc** file. For a user to create his or her own **.xinitrc** script, copying the **/etc/X11/xinit/xinitrc** first to the home directory and naming it **.xinitrc** is best. Then each user can modify the particular **.xinitrc** file as required. (Notice the system **xinitrc** file has no preceding period in its name, whereas the home directory **.xinitrc** file set up by a user does have a preceding period.) The following example shows a simplified version of the system **xinitrc** file that starts the twm window manager and an Xterm window. System and user **.Xresources** and **.Xmodmap** files are executed first to configure the X Window System.

.xinitrc

```
#!/bin/sh
userresources=$HOME/.Xresources
usermodmap=$HOME/.Xmodmap
sysresources=/etc/X11/.Xresources
sysmodmap=/etc/X11/.Xmodmap

# merge in defaults and keymaps
```

```
if [ -f $sysresources ]; then
          xrdb -merge $sysresources
fi
if [ -f $sysmodmap ]; then
           xmodmap $sysmodmap
fi
if [ -f $userresources ]; then
           xrdb -merge $userresources
fi
if [ -f $usermodmap ]; then
           xmodmap $usermodmap
fi
# start some nice programs
          xterm &
          exec twm &
```

CHAPTER 8

GNOME

The GNU Network Object Model Environment, also known as *GNOME,* is a powerful and easy-to-use environment consisting primarily of a panel, a desktop, and a set of GUI tools with which program interfaces can be constructed. GNOME is designed to provide a flexible platform for the development of powerful applications. Currently, GNOME is supported by several distributions and is the primary interface for Red Hat and Fedora. GNOME is free and released under the GNU Public License. You can download the source code, as well as documentation and other GNOME software, directly from the GNOME website at **gnome.org**. Several companies have joined together to form the GNOME Foundation, an organization dedicated to coordinating the development of GNOME and GNOME software applications. These include such companies as Sun, IBM, and Hewlett-Packard as well as Linux distributors such as Fedora, SUSE, and TurboLinux. Modeled on the Apache Software Foundation, which developed the Apache web server, the GNOME Foundation will provide direction to GNOME development as well as organizational, financial, and legal support.

The core components of the GNOME desktop consist of a panel for starting programs and desktop functionality. Other components normally found in a desktop, such as a file manager, a web browser, and a window manager, are provided by GNOME-compliant applications. GNOME provides libraries of GNOME GUI tools that developers can use to create GNOME applications. Programs that use buttons, menus, and windows that adhere to a GNOME standard can be said to be GNOME-compliant. The official file manager for the GNOME desktop is Nautilus. The GNOME desktop does not have its own window manager as KDE does. Instead, it uses any GNOME-compliant window manager. The Metacity window manager is the one bundled with the GNOME distribution.

Support for component model interfaces is integrated into GNOME, allowing software components to interconnect regardless of the computer language in which they are implemented or the kind of machine on which they are running. The standard used in GNOME for such interfaces is the Common Object Request Broker Architecture (CORBA), developed by the Object Model Group for use on Unix systems. GNOME uses the ORBit implementation of CORBA. With such a framework, GNOME applications and clients can directly communicate with each other, enabling you to use components of one application in another. With GNOME 2.0, GNOME officially adopted GConf and its libraries as the underlying method for configuring GNOME and its applications. GConf can configure independently coordinating programs such as those that make up the Nautilus file manager.

Website	Description
gnome.org	Official GNOME website
developer.gnome.org	GNOME developer website
art.gnome.org	Desktop themes and background art
gnomefiles.org	GNOME software applications, applets, and tools
gnome.org/gnome-office	GNOME office applications

Table 8-1 GNOME Resources

You can find out more about GNOME at its website, **gnome.org**. The website provides online documentation, such as the GNOME User's Guide and FAQs and also maintains extensive mailing lists for GNOME projects to which you can subscribe. The **gnomefiles.org** site provides a detailed software listing of current GNOME applications and projects. If you want to develop GNOME programs, check the GNOME developer's website at **developer.gnome.org.** The site provides tutorials, programming guides, and development tools. Here you can find the complete API reference manual online, as well as extensive support tools such as tutorials and integrated development environments (IDEs). The site also includes detailed online documentation for the GTK+ library, GNOME widgets, and the GNOME desktop. Table 8-1 offers a listing of useful GNOME sites.

GNOME 2.*x* Features

Check **gnome.org** for a detailed description of GNOME features and enhancements, with screen shots and references. GNOME releases new revisions on a frequent schedule. Several versions since the 2.0 release have added many new capabilities. Many applications and applets like Deskbar and GConf are not installed by default.

GNOME features include interface changes to Evolution, GNOME meeting, and Eye of GNOME, as well as efficiencies in load time and memory use, making for a faster response time. Gedit has been reworked to adhere to the Multiple Documentation Interface specs. New tools like F-Spot image and camera managers and the Beagle search tool are emphasized (both are .NET Mono–supported packages). The new menu editor, Alacarte, lets you customize your menus easily. The disk usage analyzer, Baobab, lets you quickly see how much disk space is used. The GNOME video player, Totem, supports web access, featuring Windows Media Player support.

The desktop images are based on Cairo, with more intuitive and user friendly icons. Buttons and windows are easier to use and appear more pleasing to the eye. The Cairo images theme is compliant with the TANGO style guidelines; TANGO is an open source standard for desktop images, providing the same image style across all open source desktops. See **tango.freedesktop.org** for more information. In addition, GNOME also adheres to the **freedesktop.org** standard naming specifications. In fact, KDE, GNOME, and XFCE all adhere to the naming specifications, using the same standard names for icons on their desktops.

For GPG encryption, signing, and decryption of files and text, GNOME provides the Seahorse Encryption Key Manager, accessible from the System menu as the Encryption

Preferences entry. With Seahorse you can manage your encryption keys stored in the GNOME keyring as well as OpenPGP SSH keys and passphrases. You can import existing keys, search for remote keys, and create your own keys. Default keyservers are listed on the Key Servers panel, to which you can add new ones. Plug-ins are provided for the gedit editor to encrypt text files, the Epiphany web browser for text phrases, and Nautilus to perform encryption from the context menu. A panel applet lets you encrypt, sign, and decrypt clipboard content.

The GNOME Control Center provides an intuitive organization and access for your desktop configuration. This is integrated into the desktop as submenus in the System | Preferences menu. Preferences are organized into Personal, Look and Feel, Internet and Network, Hardware, and System categories. The GNOME Control Center is also implemented as a GUI that will display a dialog with icons on the left for the different categories like Personal and Hardware, and a continuous list of preferences on the right. Selecting a category moves to and highlights the appropriate preferences. You can invoke the Control Center GUI by entering **`gnome-control-center`** in a terminal window.

GTK+

GTK+ is the widget set used for GNOME applications. Its look and feel was originally derived from Motif. The widget set is designed from the ground up for power and flexibility. For example, buttons can have labels, images, or any combination thereof. Objects can be dynamically queried and modified at runtime. GTK+ also includes a theme engine that enables users to change the look and feel of applications using these widgets. At the same time, the GTK+ widget set remains small and efficient.

The GTK+ widget set is entirely free under the Lesser General Public License (LGPL). The LGPL enables developers to use the widget set with proprietary software, as well as free software (the GPL would restrict it to just free software). The widget set also features an extensive set of programming language bindings, including C++, Perl, Python, Pascal, Objective C, Guile, and Ada. Internalization is fully supported, permitting GTK+-based applications to be used with other character sets, such as those in Asian languages. The drag-and-drop functionality supports drag-and-drop operations with other widget sets that support these protocols, such as Qt.

The GNOME Interface

The GNOME interface consists of the panel and a desktop, as shown in Figure 8-1. The panel appears as a long bar across the bottom of the screen. It holds menus, programs, and applets. (An *applet* is a small program designed to be run within the panel.) On the top panel is a menu labeled Applications. The menu operates like the Start menu and lists entries for applications you can run on your desktop. You can display panels horizontally or vertically and have them automatically hide to show you a full screen. The Applications menu is reserved for applications. Other tasks, such as opening a home directory window or logging out, are located in the Places menu. The System menu holds the Preferences menu for configuring your GNOME interface, as well as the Administration menu for accessing the distribution administrative tools.

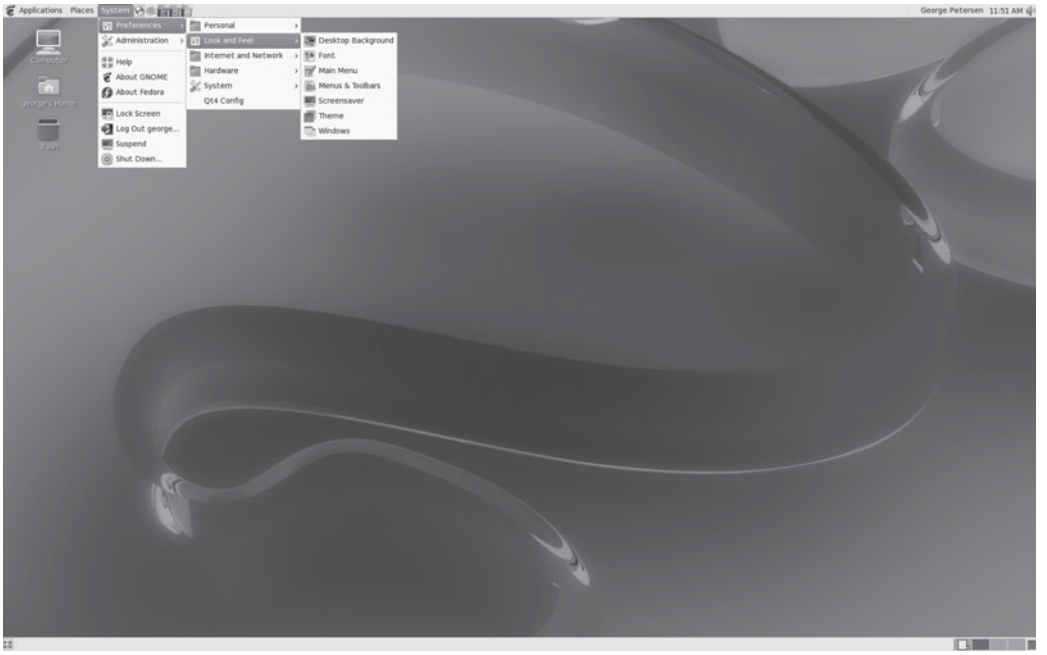


FIGURE 8-1 GNOME with Preferences menu

NOTE *The GNOME interface uses two panels, one on top for menus and notification tasks like your clock, and one on the bottom for interactive features for workspaces and docking applications. Three main menus are now used instead of one: an Applications menu, a Places menu, and the System. The System menu is used to log out of your session.*

The remainder of the screen is the desktop. Here, you can place directories, files, or programs. You can create them on the desktop directly or drag them from a file manager window. A click-and-drag operation will move a file from one window to another or to the desktop. A click and drag with the CTRL key held down will copy a file. A click-and-drag operation with the middle mouse button (two buttons at once on a two-button mouse) enables you to create links on the desktop to installed programs. Initially, the desktop holds only an icon for your home directory. Clicking it opens a file manager window to that directory. A right-click anywhere on the desktop displays a desktop menu with which you can open new windows and create new folders.

TIP *You can display your GNOME desktop using different themes that change the appearance of desktop objects such as windows, buttons, and scroll bars. GNOME functionality is not affected in any way. You can choose from a variety of themes. Many are posted on the Internet at* **art.gnome.org**. *Technically referred to as GTK themes, these allow the GTK widget set to change its look and feel. To select a theme, select Theme in the Preferences | Look And Feel menu. The default GNOME theme is Clearlooks.*

GNOME Components

From a user's point of view, you can think of the GNOME interface as having four components: the desktop, the panels, the main menus, and the file manager.

In its standard default configuration, the GNOME desktop displays a Folder icon for your home directory in the upper-left corner, along with a trash can to delete items. In addition, the desktop also displays a Computer window for accessing the entire file system, CD/DVD drives, and network shares. Double-clicking the home directory icon will open the file manager, displaying files in your home directory. You have two panels displayed, one used for menus, application icons, and running applets at the top of the screen, and one at the bottom of the screen used primarily for managing your windows and desktop spaces.

The top bar has several menus and application icons: the Applications menu, the Places menu, the System menu, the Mozilla Firefox web browser (globe with fox), and the Evolution mail tool (envelope). To the right are the time and date icons. An update button will appear if updates are available. You can use the update icon to automatically update your system. The bottom bar holds icons for minimized windows as well as running applets. These include a Workspace Switcher (squares) placed to the right. An icon to the left lets you minimize all your open windows. When you open a window, a corresponding button for it will be displayed in the lower panel, which you can use to minimize and restore the window.

To start a program, you can select its entry in the Applications menu. You can also click its application icon in the panel (if there is one) or drag a data file to its icon.

Quitting GNOME

To quit GNOME, you select the Logout or Shutdown entries in the System menu. The Logout entry quits GNOME, returning you to the login window (or command line shell still logged in to your Linux account, if you started GNOME with **`startx`**). The Shut Down entry displays a dialog that allows you to hibernate, shut down, cancel, or restart your system. A Restart entry shuts down and reboots your system. You must separately quit a window manager that is not GNOME-compliant after logging out of GNOME.

GNOME Help

The GNOME Help browser (Yelp) provides a browserlike interface for displaying the GNOME user's manual, Man pages, and info documents. You can select it from the System menu. It features a toolbar that enables you to move through the list of previously viewed documents. You can even bookmark specific items. A browser interface enables you to use links to connect to different documents. On the main page, expandable links for several GNOME desktop topics are displayed on left side, with entries for the GNOME User Manual and Administration Guide on the right side. At the bottom of the left side listing are links for the Man and Info pages. You can use these links to display Man and Info pages easily. Use the Search box to quickly locate help documents. Special URL-like protocols are supported for the different types of documents: **ghelp**, for GNOME help; **man**, for Man pages; and **info**, for the info documents, such as **man:fstab** to display the Man page for the **fstab** file.

The GNOME Help browser provides a detailed manual on every aspect of your GNOME interface. The left-hand links display GNOME categories for different application categories such as the System tools and GNOME applets. The GNOME Applets entry

provides detailed descriptions of all available GNOME applets. Applications categories like Internet, Programming, System Tools, and Sound and Video will provide help documents for applications developed as part of the GNOME project, like the Evolution mail client, the Totem movie player, the Disk Usage Analyzer, and the GNOME System Monitor. Click the Desktop entry at the top of the left hand list to display links for the GNOME User and Administration manuals.

The GNOME Desktop

The GNOME desktop provides you with all the capabilities of GUI-based operating systems (refer to Figure 8-1). You can drag files, applications, and directories to the desktop, and then back to GNOME-compliant applications. If the desktop stops functioning, you can restart it by starting the GNOME file manager (Nautilus). The desktop is actually a back-end process in the GNOME file manager, but you needn't have the file manager open to use the desktop.

NOTE *As an alternative to using the desktop, you can drag any program, file, or directory to the panel and use the panel instead.*

Drag and Drop Files to the Desktop

Any icon for an item that you drag from a file manager window to the desktop also appears on the desktop. However, the default drag-and-drop operation is a **`move`** operation. If you select a file in your file manager window and drag it to the desktop, you are actually moving the file from its current directory to the GNOME desktop directory, which is located in your home directory and holds all items on the desktop. For GNOME, the desktop directory is **DESKTOP**. In the case of dragging directory folders to the desktop, the entire directory and its subdirectories will be moved to the GNOME desktop directory. To remove an icon from the desktop, you move it to the trash.

You can also copy a file to your desktop by pressing the CTRL key and then clicking and dragging it from a file manager window to your desktop. You will see the small arrow in the upper-right corner of the copied icon change to a + symbol, indicating that you are creating a copy, instead of moving the original.

CAUTION *Be careful when removing icons from the desktop. If you have moved the file to the desktop, then its original is residing in the DESKTOP folder, and when you remove it you are erasing the original. If you have copied or linked the original, then you are simply deleting the link or the copy. When you drag applications from a menu or panel to the desktop, you are just creating a copy of the application launcher button in the DESKTOP directory. These you can safely remove.*

You can also create a link on the desktop to any file. This is useful if you want to keep a single version in a specified directory and be able to access it from the desktop. You can also use links for customized programs that you may not want on a menu or panel. There are two ways to create a link. While holding down the CTRL and SHIFT keys (CTRL-SHIFT), drag the file to where you want the link created. A copy of the icon then appears with a small arrow

in the right corner, indicating it is a link. You can click this link to start the program, open the file, or open the directory, depending on what kind of file you linked to. Alternatively, first click and drag the file out of the window, and after moving the file but before lifting up the mouse button, press the ALT key. This will display a pop-up menu with selections for Cut, Copy, and Link. Select the Link option to create a link.

GNOME's drag-and-drop file operation works on virtual desktops provided by the GNOME Workspace Switcher. The GNOME Workspace Switcher on the bottom panel creates icons for each virtual desktop in the panel, along with task buttons for any applications open on them.

NOTE *Although the GNOME desktop supports drag-and-drop operations, these normally work only for applications that are GNOME-compliant. You can drag any items from a GNOME-compliant application to your desktop, and vice versa.*

Applications on the Desktop

In most cases, you only want to create on the desktop another way to access a file without moving it from its original directory. You can do this either by using a GNOME application launcher button or by creating a link to the original program. Application launcher buttons are the GNOME components used in menus and panels to display and access applications. The Open Office buttons on the top panel are application launcher buttons. To place an icon for the application on your desktop, you can simply drag the application button from the panel or from a menu. For example, to place an icon for the Firefox web browser on your desktop, just drag the web browser icon on the top panel to anywhere on your desktop space.

For applications that are not on a panel or in a menu, you can create either an application launcher button for it or a direct link, as described in the preceding section. To create an application launcher, first right-click the desktop background to display the desktop menu. Then select the Create Launcher entry.

GNOME Desktop Menu

You can also right-click anywhere on the empty desktop to display the GNOME desktop menu. This will list entries for common tasks, such as creating an application launcher, creating a new folder, or organizing the icon display. Keep in mind that the New Folder entry creates a new directory on your desktop, specifically in your GNOME desktop directory (**DESKTOP**), not your home directory. The entries for this menu are listed in Table 8-2.

Window Manager

GNOME works with any window manager. However, desktop functionality, such as drag-and-drop capabilities and the GNOME Workspace Switcher (discussed later), works only with window managers that are GNOME-compliant. The current release of GNOME uses the Metacity window manager. It is completely GNOME-compliant and is designed to integrate with the GNOME desktop without any duplication of functionality. Other window managers such as Enlightenment, IceWM, and Window Maker can also be used. Check a window manager's documentation to see if it is GNOME-compliant.

For 3-D support you can use compositing window managers like Compdiz or Beryl. Windows are displayed using window decorators, allowing windows to wobble, bend, and

Menu Item	Description
Create Launcher	Creates a new desktop icon for an application.
Create Folder	Creates a new directory on your desktop within your **DESKTOP** directory.
Create Document	Creates files using installed templates.
Clean Up by Name	Arranges your desktop icons.
Keep Aligned	Aligns your desktop icons.
Cut, Copy, Paste	Cuts, copies, or pastes files, letting you move or copy files between folders.
Change Desktop Background	Opens a Background Preferences dialog to let you select a new background for your desktop.

TABLE 8-2 The GNOME Desktop Menu

move in unusual ways. They employ features similar to current Mac and Vista desktops. A compositing window manager relies on a graphics card OpenGL 3-D acceleration support. Be sure your graphics card is supported. Compiz may be installed on your distribution as the default 3-D support. See **compiz.org** for more information. Beryl was developed from Compiz and features its own window decorators. You can find more about Beryl at **beryl-project.org**. Both projects plan to merge, providing a single compositing window manager for Linux.

Metacity employs much the same window operations as used on other window managers. You can resize a window by clicking any of its sides or corners and dragging. You can move the window with a click-and-drag operation on its title bar. You can also right-click and drag any border to move the window, as well as ALT-click anywhere on the window. The upper-right corner shows the Maximize, Minimize, and Close buttons. Minimize creates a button for the window in the panel that you can click to restore it. You can right-click the title bar of a window to display a window menu with entries for window operations. These include workspace entries to move the window to another workspace (virtual desktop) or to all workspaces, which displays the window no matter to what workspace you move.

The GNOME Volume Manager

Managing DVDs/CD-ROMs, card readers, floppy disks, digital cameras, and other removable media is the task of the GNOME Volume Manager. This is a lower-level utility that remains transparent to the user, though how you treat removable media can be configured with the Drives and Removable Media preferences tool. The GNOME Volume Manager allows you not only to access removable media, but also to access all your mounted file systems, remote and local, including any Windows shared directories accessible from Samba. You can browse all your file systems directly from GNOME, which implements this capability with the gnome virtual file system (gnome-vfs) mapping to your drives, storage devices, and removable media. The GNOME Volume Manager uses HAL and udev to access removable media directly, and Samba to provide Windows networking support. Media are mounted by

gnomemount, a wrapper for accessing HAL and udev, which perform the mount (**/etc/fstab** is no longer used).

You can access your file systems and removable media using the Computer icon on the desktop. This opens a top-level window showing icons for all removable media (mounted CD-ROMs, floppies, and so on), your local file system, and your network shared resources (see Figure 8-2). Double-click any icon to open a file manager window, displaying its contents. The file system icon will open a window showing the root-level directory, the top directory for your file system. Access will be restricted for system directories, unless you log in as the root user. The network icon will open a window listing your connected network hosts. Opening these will display the shares, such as shared directories, that you can have access to. Drag-and-drop operations are supported for all shared directories, letting you copy files and folders from a shared directory on another host to a directory on your system. To browse Windows systems on GNOME using Samba, you first have to configure your firewall to accept Samba connections.

Removable media will also appear automatically as icons directly on your desktop. A DVD or CD-ROM is automatically mounted when you insert it into your DVD/CD-ROM drive, displaying an icon for it with its label. The same kind of access is also provided for card readers, digital cameras, and USB drives. Be sure to unmount the USB drives before removing them so that data will be written.

You can then access the disc in the DVD/CD-ROM drive either by double-clicking it or by right-clicking and selecting the Open entry. A file manager window opens to display the contents of the CD-ROM disc. To eject a CD-ROM, you can right-click its icon and select Eject from the pop-up menu. The same procedure works for floppy disks, using the Floppy Disk icon. Be sure you don't remove a mounted floppy disk until you have first unmounted it, selecting the Eject entry in the pop-up menu.

Burning a data DVD/CD is a simple matter of placing a blank DVD in your drive. Nautilus automatically recognizes it as a blank disc and allows you to write to it. All read/write discs, even if they are not blank, are also recognized as writable discs and opened up in a DVD/CD writer window. To burn a disc, just drag the files you want to copy to the blank disc window and then click Write To Disc. A dialog will open up with buttons to set options like the write speed and disc label. After writing, a dialog then lists buttons to eject, burn again, or close. Keep in mind that the newly written disc is not mounted. You can eject it at any time.

Nautilus can also burn ISO DVD and CD images. Just insert a blank DVD or CD and then drag the ISO disc image file to a blank CD/DVD icon on your desktop. A dialog will open up asking you if you want to burn the DVD or CD image. Nautilus works with ISO images, that is, files ending with a **.iso** suffix. For other image files such as IMG files, you can change the suffix to **.iso,** and Nautilus will recognize and burn the image file normally.

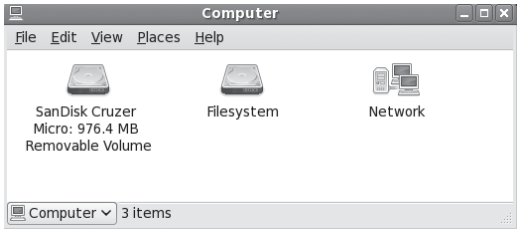


FIGURE 8-2 GNOME Computer window (GNOME Volume Manager).

GNOME will display icons for any removable media and perform certain default actions on them. For example, audio CDs will be automatically played in the CD player. DVD movies can be started up in a DVD player. To set the preferences for how removable media are treated, you use the Drives and Removable Media preferences tool, accessible with the Removable Media entry in the System | Preferences | Hardware menu. Certain settings are already set.

NOTE *GNOME now manages all removable media directly with HAL, instead of using fstab entries.*

The GNOME File Manager: Nautilus

Nautilus is the GNOME file manager, supporting the standard features for copying, removing, and deleting items, as well as setting permissions and displaying items. It also provides enhancements such as zooming capabilities, user levels, and theme support. You can enlarge or reduce the size of your file icons; select from novice, intermediate, or expert levels of use; and customize the look and feel of Nautilus with different themes. Nautilus also lets you set up customized views of file listings, enabling you to display images for directory icons and run component applications within the file manager window. Nautilus implements a spatial approach to file browsing. A new window is opened for each new folder.

Nautilus Window

Nautilus was designed as a desktop shell in which different components can be employed to add functionality. For example, within Nautilus, a web browser can be executed to provide web browser capabilities in a Nautilus file manager window. An image viewer can display images. The GNOME media player can run sound and video files. The GNOME File Roller tool can archive files, as well as extract them from archives. With the implementation of GStreamer, multimedia tools such as the GNOME audio recorder are now more easily integrated into Nautilus.

TIP *Several distributions such as Red Hat and Fedora use the Common User Directory Structure (xdg-user-dirs at* **freedesktop.org***) to set up subdirectories such as* **Music** *and* **Video** *in the user home directory. These localized user directories are used as defaults by many desktop applications. Users can change their directory names or place them within each other using the GNOME file browser. For example, Music can be moved into* **Documents**, **Documents/Music**. *Local configuration is held in the* **.config/user-dirs.dirs** *file. Systemwide defaults are set up in the* **/etc/xdg/user-dirs.defaults** *file.*

By default, the Nautilus windows are displayed with the Spatial view. This provides a streamlined display with no toolbars or sidebar (see Figure 8-3). Much of its functionality has been moved to menus and pop-up windows, leaving more space to display files and folders. You can, however, open a Nautilus window in the Browser view, which will display the traditional menu bar and location toolbars. You can open a window in the Browser view by right-clicking the folder icon and selecting Browse Folder from the pop-up menu.

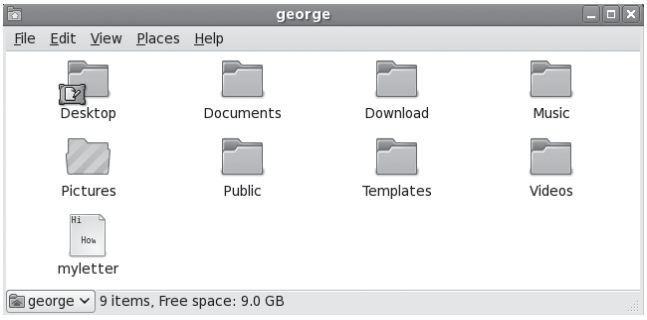


FIGURE 8-3 Spatial view, Nautilus window

The Spatial view of a Nautilus window displays a menu bar at the top with menus for managing your files. An information bar at the bottom displays information about the directory or selected files. To the lower left is a pop-up window displaying the parent directories for your current working directory. You can select any entry to open a window for that directory.

With the Browser view, a Nautilus window displays toolbars, including a menu bar of file manager commands and a Location toolbar at the top which can toggle between a location box or button views (see Figure 8-4), along with a sidebar for file and directory information. The rest of the window is divided into two panes. The left pane is a side pane used to display information about the current working directory. The right pane is the main panel that displays the list of files and subdirectories in the current working directory. A status bar at the bottom of the window displays information about a selected file or directory. You can turn any of these elements on or off by selecting their entries in the View menu.

Next to the Location bar (box or button) is an element for zooming in and out of the view of the files. Click the + button to zoom in and the – button to zoom out. Next to the zoom element is a drop-down menu for selecting the different views for your files, such as icons, small icons, or details.

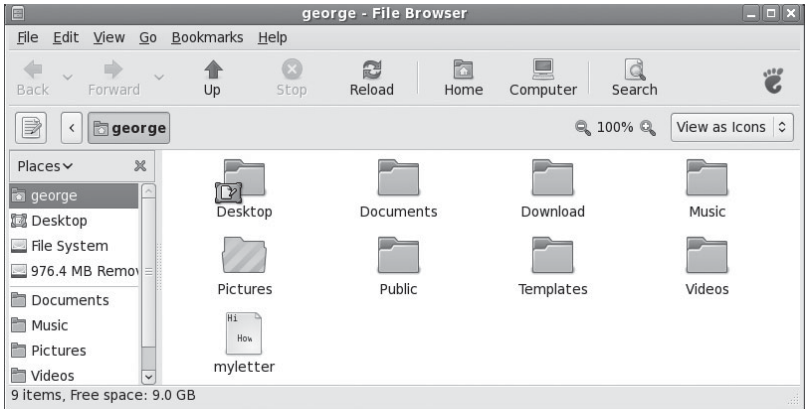


FIGURE 8-4 Browser view, Nautilus file manager window

NOTE *Nautilus features built-in DVD/CD-burning support with the nautilus-cd-burner package for both files and ISO images.*

Nautilus Sidebar: Tree, History, and Notes

The sidebar has several different views, selectable from a pop-up menu, for displaying additional information about files and directories: Places, Information, Tree, History, and Notes. The Places view shows your file system locations that you would normally access, starting with your home directory. File System places you at top of the file system, letting you move to any accessible part of it. Information displays detailed information about the current directory or selected file. For example, if you double-click an image file, the Information pane will display detailed data on the image, while the Window pane displays the full image. The Tree view displays a tree-based hierarchical view of the directories and files on your system, highlighting the one you have currently selected. You can use this tree to move to other directories and files. The tree maps all the directories on your system, starting from the root directory. You can expand or shrink any directory by clicking the + or – symbol before its name. Select a directory by clicking the directory name. The contents of that directory are then displayed in the main panel. The History view shows previous files or directories you have accessed, handy for moving back and forth between directories or files.

The Notes view displays notes you have entered about an item or directory. The Notes view opens an editable text window within the side pane. Just select the Notes view and type in your notes. To add a note for a particular item, such as an image or sound file, just double-click the item to display or run it, and then select the Note view to type in your note. You can also right-click the item, to display the item's pop-up menu and select preferences, from which you can click a Notes panel. After you have added a note, you will see a note image added to the item's icon in the Nautilus window.

Displaying Files and Folders

You can view a directory's contents as icons or as a detailed list. In the Spatial view, you select the different options from the View menu. In the Browser view, you use the pop-up menu located on the right side of the Location bar. The List view provides the name, permissions, size, date, owner, and group. In the View as List view, buttons are displayed for each field across the top of the main panel. You can use these buttons to sort the lists according to that field. For example, to sort the files by date, click the Date button; to sort by size, click Size.

In the Icon view, you can sort icons and preview their contents without opening them. To sort items in the Icon view, select the Arrange Items entry in the View menu (Spatial or Browser view) and then select a layout option. Certain types of file icons will display previews of their contents—for example, the icons for image files will display a small version of the image. A text file will display in its icon the first few words of its text. The Zoom In entry enlarges your view of the window, making icons bigger, and Zoom Out reduces your view, making them smaller. Normal Size restores them to the standard size. You can also use the + and – buttons on the Location bar to change sizes.

In both the Spatial and Browser views, you can also change the size of individual icons. Select the icon and then choose the Stretch entry from the Edit menu. Handles will appear on the icon image. Click and drag the handles to change its size. To restore the icon, select Restore Icon's Original Size in the Edit menu.

To add an emblem to any file or directory icon, just select the Background & Emblems entry from the Edit menu to open the Background & Emblems window. Here you will see three icons to display panels for color and pattern backgrounds, as well as file and directory emblems. Click one of the emblems to display the selection of emblems. To add an emblem to a file or directory icon, click and drag the emblem from the Emblem panel to the file or directory icon. The emblem will appear on that icon. If you want to add your own emblem, click the Add Emblem button to search for an emblem image file by name, or browse your file system for the image you want to use (click the Image icon).

Nautilus Menu

You can click anywhere on the main panel to display a pop-up menu with entries for managing and arranging your file manager icons (see Table 8-3). The menu is the same for both Spatial and Browser views. To create a new folder, select Create Folder. The Arrange Items entry displays a submenu with entries for sorting your icons by name, size, type, date, or even emblem. The Manually entry lets you move icons wherever you want on the main panel. You can also cut, copy, and paste files to more easily move or copy them between folders.

TIP *To change the background used on the File Manager window, you select Background & Emblems from the Edit menu, dragging the background you want to the file manager window. Choose from either colors or patterns.*

Navigating Directories

The Spatial and Browser views use different tools for navigating directories. The Spatial view relies more on direct window operations, whereas the Browser view works more like a browser. Recall that to open a directory with the Browser view, you need to right-click the directory icon and select Browse Folder.

Menu Item	Description
Create Folder	Creates a new subdirectory in the directory.
Create Document	Creates a new document using installed templates.
Arrange Items	Displays a submenu to arrange files by name, size, type, date, or emblem.
Cut, Copy, Paste	Cuts, copies, or pastes files, letting you move or copy files between folders.
Zoom In	Provides a close-up view of icons, making them appear larger.
Zoom Out	Provides a distant view of icons, making them appear smaller.
Normal Size	Restores view of icons to standard size.
Properties	Opens the Properties panels for the directory opened in the window.
Clean Up by Name	Arranges icons by name.

TABLE 8-3 Nautilus File Manager Menu

Navigating in the Spatial View

In the Spatial view, Nautilus will open a new window for each directory selected. To open a directory, either double-click it or right-click and select the Open entry. The parent directory pop-up menu at the bottom left lets you open a window for any parent directories, in effect, moving to a previous directory. To jump to a specific directory, select the Open Location entry from the File menu. This will, of course, open a new window for that directory. The Open Parent entry on the File menu lets you quickly open a new window for your parent. You will quickly find that moving to different directories entails opening many new windows.

Navigating in the Browser View

The Browser view of the Nautilus file manager operates similarly to a web browser, using the same window to display opened directories. It maintains a list of previously viewed directories, and you can move back and forth through that list using the toolbar buttons. The LEFT ARROW button moves you to the previously displayed directory, and the RIGHT ARROW button moves you to the next displayed directory. The UP ARROW button moves you to the parent directory, and the HOME button moves you to your home directory. To use a pathname to go directly to a given directory, you can type the pathname in the Location box and press ENTER. Use the toggle icon at the left of the location bar to toggle between box and button location views.

To open a subdirectory, you can double-click its icon or single-click the icon and select Open from the File menu. If you want to open a separate Nautilus Browser view window for that directory, right-click the directory's icon and select Open In A New Window.

Managing Files

As a GNOME-compliant file manager, Nautilus supports GUI drag-and-drop operations for copying and moving files. To move a file or directory, click and drag from one directory to another as you would on Windows or Mac interfaces. The move operation is the default drag-and-drop operation in GNOME. To copy a file, click and drag normally while pressing the CTRL key.

NOTE *If you move a file to a directory on another partition (file system), it will be copied instead of moved.*

The File Menu

You can also perform remove, rename, and link-creation operations on a file by right-clicking its icon and selecting the action you want from the pop-up menu that appears (see Table 8-4). For example, to remove an item, right-click it and select the Move To Trash entry from the pop-up menu. This places it in the **Trash** directory, where you can later delete it by selecting Empty Trash from the Nautilus File menu. To create a link, right-click the file and select Make Link from the pop-up menu. This creates a new link file that begins with the term "link."

Renaming Files

To rename a file, you can right-click the file's icon and select the Rename entry from the pop-up menu (or just press the R key). The name of the icon will be highlighted in a black background, encased in a small text box. You then click the name and delete the old name

Menu Item	Description
Open	Opens a file with its associated application. Directories are opened in the file manager. Associated applications will be listed.
Open In A New Window	Opens a file or directory in a separate window. Browser view only.
Open With Other Application	Selects an application with which to open a file. A submenu of possible applications is displayed.
Cut, Copy, Paste files	Entries to cut, copy, or paste files.
Make Link	Creates a link to a file in the same directory.
Rename	Renames a file.
Move To Trash	Moves a file to the **Trash** directory, where you can later delete it.
Create Archive	Archives a file using File Roller.
Send To	E-mails a file.
Properties	Displays the Properties dialog box for a file. There are three panels: Statistics, Options, and Permissions.

TABLE 8-4 The Nautilus File Pop-Up Menu

by typing a new one. You can also rename a file by entering a new name in its Properties dialog box. Right-click and select Properties from the pop-up menu to display the Properties dialog box. On the Basic tab, change the name of the file.

File Grouping

File operations can be performed on a selected group of files and directories. You can select a group of items in several ways. You can click the first item and then hold down the SHIFT key while clicking the last item. You can also click and drag the mouse across items you want to select. To select separated items, hold the CTRL key down as you click the individual icons. If you want to select all the items in the directory, choose the Select All entry in the Edit menu. You can then click and drag a set of items at once. This enables you to copy, move, or even delete several files at once.

Applications and Files: Open With

You can start any application in the file manager by double-clicking either the application itself or a data file used for that application. If you want to open the file with a specific application, right-click the file and select the Open With Other Application entry. A submenu displays a list of possible applications. If your application is not listed, select Other Application to open a Select An Application dialog box, where you can choose the application with which you want to open this file. You can also use a text viewer to display the bare contents of a file within the file manager window. Drag-and-drop operations are also supported for applications. You can drag a data file to its associated application icon (say, one on the desktop); the application then starts up using that data file.

To change or set the default application to use for a certain type of file, open a file's Properties and select the Open With panel. Here you can choose the default application to use for that kind of file. For example, changing the default for an image file from Image Viewer to KView will make KView the default viewer for all image files. If the application you want is

not listed, click the Add button in the Open With panel to display a listing of applications and choose the one you want. This displays an Add Application box and a Browse button. Commonly used applications are already listed. If you already know the full pathname of the application, you can enter it directly. If the application is not listed, click Browse to display a Select An Application box that will list applications to choose from. Initially, applications in the **/usr/bin** directory are listed, though you can browse to other directories. Once you select your application, it will appear in the Open With list for this file.

If there is an application on the Open With panel you do not want listed in the Open With options, select it and click the Remove button.

For example, to associate BitTorrent files with the original BitTorrent application, right-click any BitTorrent file (one with a **.torrent** extension), select the Properties entry, and then select the Open With panel. A list of installed applications will be displayed, such as Ktorrent, Azureus, and BitTorrent. Click BitTorrent to use the original BitTorrent application, then close. BitTorrent will then be the default for **.torrent** files.

TIP *The Preferred Applications tool will let you set default applications for Internet and system applications, namely the web browser, mail client, and terminal window console. Available applications are listed in pop-up menus. You can even select from a list of installed applications for select a custom program. You access the Preferred Applications tool from the Personal submenu located in the System | Preferences menu.*

Application Launcher

Certain files, such as shell scripts, are meant to be executed as applications. To run the file using an icon as you would other installed applications, you can create an application launcher for it. You can create application launchers using the Create Launcher tool. This tool is accessible either from the desktop menu as the Create Launcher entry, or from the panel menu's Add To box as the Custom Application Launcher entry. When created from the desktop, the new launcher is placed on the desktop; when created from a panel, it will be placed directly on that panel.

The Create Launcher tool will prompt you for the application name, the command that invokes it, and the launch type. For the launch type you have the choice of application, file, or file within a terminal. For shell scripts, you use an Application In Terminal option, running the script within a shell.

Use the file type for a data file for which an associated application will be automatically started, opening the file—for example, a web page—which will then start a web browser. Instead of a command, you will be prompted to enter the location of the file.

For Applications and Applications In Terminal, you will be prompted to select the command to use. To do this (the actual application or script file), you can either enter its pathname, if you know it, or use the Browse button to open a file browser window to select it.

To select an icon for your launcher, click the Icon button, initially labeled No Icon. This opens the Icon Browser window, listing icons from which you can choose.

File and Directory Properties

With the Properties dialog box, you can view detailed information on a file, and set options and permissions (see Figure 8-5). A Properties box has five panels: Basic, Emblems, Permissions, Open With, and Notes. The Basic panel shows detailed information such as type, size, location,

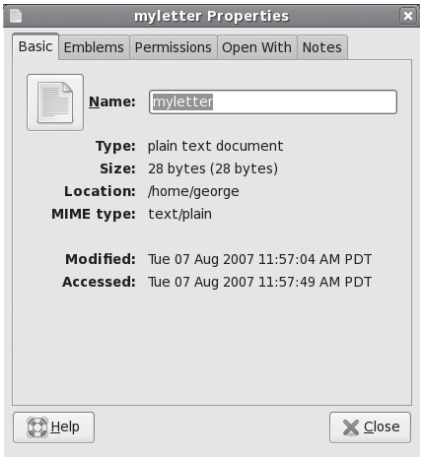


FIGURE 8-5
File properties on Nautilus.

and date modified. The type is MIME, indicating the type of application associated with it. The file's icon is displayed at the top, and you can edit the filename in the text box under the icon. If you want to change the icon image used for the file or folder, click the icon image on the Basic panel (next to the name). A Select Custom Icon dialog will open, showing available icons; select the one you want. The **pixmaps** directory holds the set of current default images, though you can select your own images also. Click the Image entry to see its icon displayed in the right panel. Double-clicking effects the icon image change.

The Emblems panel enables you to set the emblem you want displayed for this file, displaying all the emblems available. An emblem will appear in the upper-right corner of the icon, giving an indication of the file's contents or importance.

The Permissions panel shows the read, write, and execute permissions for owner, group, and other, as set for the file. You can change any of the permissions here, provided the file belongs to you. You configure access for owner, group, and others, using pop-up menus. You can set owner permissions as Read Only or Read And Write. For the group and others, you can also set the None option, denying access. The group name expands to a pop-up menu listing different groups; select one to change the file's group. If you want to execute this as an application (say, a shell script), check the Allow Executing File As Program entry. This has the effect of setting the execute permission.

The Permissions panel for directories operates much the same way, but it includes two access entries, Folder Access and File Access. The Folder Access entry controls access to the folder with options for List Files Only, Access Files, and Create And Delete Files. These correspond to the read, read and execute, and read/write/execute permissions given to directories. The File Access entry lets you set permissions for all files in the directory. They are the same as for files: for the owner, Read or Read and Write; for the group and others, the entry adds a None option to deny access. To set the permissions for all the files in the directory accordingly (not just the folder), click the Apply Permissions To Enclosed Files button.

The Open With panel lists all the applications associated with this kind of file. You can select which one you want as the default. This can be particularly useful for media files, where you may prefer a specific player for a certain file, or a particular image viewer for pictures.

The Notes panel will list any notes you want to make for the file or directory. It is an editable text window, so you can change or add to your notes, directly.

Certain kind of files will have added panels, providing information about the item. For example, an audio file will have an Audio panel listing the type of audio file and any other information, such as the song title or compression method used. An image file will have an Image panel listing the resolution and type of image. A video file will contain a Video panel showing the type of video file along with compression and resolution information.

Nautilus Preferences

You can set preferences for your Nautilus file manager in the Preferences dialog box, which you can access by selecting the Preferences item in the Edit menu. The Preferences dialog box shows a main panel with a sidebar with several configuration entries, including Views, Behavior, Display, List Columns, and Preview. You use these dialog boxes to set the default display properties for your Nautilus file manager.

- The Views panel allows you to select how files are displayed by default, such as the list or icon view.
- Behavior lets you choose how to select files, manage the trash, and handle scripts, as well as whether to use the Browser view as the default.
- Display lets you choose what added information you want displayed in a icon caption, like the size or date.
- List Columns view lets you choose both the features to display in the detailed list and the order to display them in. In addition to the already-selected Name, Size, Date, and Type, you can add permissions, group, MIME type, and owner.
- The Preview panel lets you choose whether you want small preview content displayed in the icons, like beginning text for text files.

Nautilus as a FTP Browser

Nautilus works as an operational FTP browser. You can use the Location box (toggle to box view) or the Open Location entry on the File menu to access any FTP site. Just enter the URL for the FTP site in the Location box and press ENTER (you do not need to specify **ftp://**). Folders on the FTP site will be displayed, and you can drag files to a local directory to download them. The first time you connect to a site, an Authentication dialog will open, letting you select either Anonymous access or access as a User. If you select User, you can then enter your username and password for that site. You can then choose to remember the password for just this session, or permanently by storing it in a keyring.

Once you have accessed the site, you can navigate through the folders as you would with any Nautilus folder, opening directories or returning to parent directories. To download a file, just drag it from the FTP window to a local directory window. A small dialog will appear showing download progress. To upload a file, just drag it from your local folder to the window for the open FTP directory. Your file will be uploaded to that FTP site (should you have permission to do so). You can also delete files on the site's directories.

NOTE *Unlike KDE's Konqueror file manager, Nautilus is not a functional web browser. It is preferable that you use a web browser for access to the Web.*

The GNOME Panel

The *panel* is the center of the GNOME interface. Through it you can start your applications, run applets, and access desktop areas. You can think of the GNOME panel as a type of tool you can use on your desktop. You can have several GNOME panels displayed on your desktop, each with applets and menus you have placed in them. In this respect, GNOME is flexible, enabling you to configure your panels any way you want. In fact, the default GNOME desktop features two panels, a menu panel at the top for your applications and actions (see Figure 8-6), and a panel at the bottom used for minimized windows and the Workspace Switcher. You can customize a panel to fit your own needs, holding applets and menus of your own selection. You may add new panel, add applications to the panel, and add various applets.

Panel configuration tasks such as adding applications, selecting applets, setting up menus, and creating new panels are handled from the Panel pop-up menu. Just right-click anywhere on your panel to display a menu with entries for Properties, New Panel, Add To Panel, and Delete This Panel, along with Help and About entries. New Panel lets you create other panels; Add To Panel lets you add items to the panel, such as application launchers, applets for simple tasks like the Workspace Switcher, and menus like the main applications menu. The Properties entry will display a dialog for configuring the features for that panel, like the position of the panel and its hiding capabilities.

To add a new panel, select the New Panel entry in the Panel pop-up menu. A new expanded panel is automatically created and displayed on the side of your screen. You can then use the panel's Properties box to set different display and background features, as described in the following sections.

Panel Properties

To configure individual panels, you use the Panel Properties dialog box. To display this dialog box, you right-click the particular panel and select the Properties entry in the pop-up menu. For individual panels, you can set general configuration features and the background. The Panel Properties dialog box includes a tabbed pane, General and Background. With version 2.4, GNOME abandoned the different panel types in favor of just one kind of panel with different possible features that give it the same capabilities as the old panel types.

Displaying Panels

On the General pane of a panel's Properties box, you determine how you want the panel displayed. Here you have options for orientation, size, and whether to expand, auto-hide, or display hide buttons. The Orientation entry lets you select which side of the screen you want the panel placed on. You can then choose whether you want a panel expanded. An expanded panel will fill the edges of the screen, whereas a nonexpanded panel is sized to the number of items in the panel and shows handles at each end. Expanded panels will remain fixed to the edge of screen, whereas unexpanded panels can be moved, provided the Show Hide Buttons feature is not selected.



Figure 8-6 The GNOME panel at the top of the desktop

Moving and Hiding Expanded Panels

Expanded panels can be positioned at any edge of your screen. You can move expanded panels from one edge of a screen to another by simply dragging the panel to another edge. If a panel is already there, the new one will stack on top of the current one. You cannot move unexpanded panels in this way. Bear in mind that if you place an expanded panel on the side edge, any menus will be displayed across at the top corner to allow proper pop-up display. The panel on the side edge will expand in size to accommodate its menus. If you have several menus or a menu with a lengthy names, you could end up with a very large panel.

You can hide expanded panels either automatically or manually. These are features specified in the panel properties General box as Auto Hide and Show Hide Buttons. To automatically hide panels, select the Auto Hide feature. To redisplay the panel, move your mouse to the edge where the panel is located. You can enable or disable the Hide buttons in the panel's Properties window.

If you want to be able to hide a panel manually, select Show Hide Buttons. Two handles will be displayed at either end of the panel. You can further choose whether to have these handles display arrows. You can then hide the panel at any time by clicking either of the Hide buttons located on each end of the panel. The Hide buttons are thin buttons showing a small arrow. This is the direction in which the panel will hide.

Unexpanded Panels: Movable and Fixed

Whereas an expanded panel is always located at the edge of the screen, an unexpanded panel is movable. It can be located at the edge of a screen, working like a shrunken version of an expanded panel, or you can move it to any place on your desktop, just as you would an icon.

An unexpanded panel will shrink to the number of its components, showing handles at either end. You can then move the panel by dragging its handles. To access the panel menu with its Properties entry, right-click either of its handles.

To fix an unexpanded panel at its current position, select the Show Hide Buttons feature on its Properties box. This will replace the handles with Hide buttons and make the panel fixed. Clicking a Hide button will hide the panel to the edge of the screen, just as with expanded panels. If an expanded panel is already located on that edge, the button for a hidden unexpanded panel will be on top of it, just as with a hidden expanded panel. The Auto Hide feature will work for unexpanded panels placed at the edge of a screen.

If you want to fix an unexpanded panel to the edge of a screen, make sure it is placed at the edge you want, and then set its Show Hide Buttons feature.

Panel Background

With a panel's Background pane on its Properties box, you can change the panel's background color or image. For a color background, you click a color button to display a color selection window where you can choose a color from a color circle and its intensity from an inner color triangle. You can enter its number, if you know it. Once your color is selected, you can use the Style slider bar to make it more transparent or opaque. To use an image instead of a color, select the image entry and use the Browse button to locate the image file you want. For an image, you can also drag and drop an image file from the file manager to the panel; that image then becomes the background image for the panel.

Panel Objects

A panel can contain several different types of objects. These include menus, launchers, applets, drawers, and special objects.

- **Menus** The Applications menu is an example of a panel menu. Launchers are buttons used to start an application or execute a command.
- **Launchers** The web browser icon is an example of a launcher button. You can select any application entry in the Applications menu and create a launcher for it on the panel.
- **Applets** An applet is a small application designed to run within the panel. The Workspace Switcher showing the different desktops is an example of a GNOME applet.
- **Drawers** A drawer is an extension of the panel that can be open or closed. You can think of a drawer as a shrinkable part of the panel. You can add anything to it that you can to a regular panel, including applets, menus, and even other drawers.
- **Special objects** Special objects are used for special tasks not supported by other panel objects. For example, the Logout and Lock buttons are special objects.

Moving, Removing, and Locking Objects

To move any object within the panel, right-click it and choose Move Entry. You can move it either to a different place on the same panel or to a different panel. For launchers, you can just drag the object directly where you want it to be. To remove an object from the panel, right-click it to display a pop-up menu for it, and then choose the Remove From Panel entry. To prevent an object from being moved or removed, you set its lock feature (right-click the object and select the Lock entry). To later allow it to be moved, you first have to unlock the object (right-click it and select Unlock).

Tip *On the panel Add To list, common objects like the clock and the CD player are intermixed with object types like menus and applications. When adding a kind of object, such as an application, you will have to search through the list to find the entry for that type; in the case of applications, it is the Application Launcher entry.*

Adding Objects

To add an object to a panel, select the object from the panel's Add To box (see Figure 8-7). To display the Add To box, right-click the panel and select the Add To Panel entry. This Add To box displays a lengthy list of common objects as well as object types. For example, it will display the Main menu as well as an entry for creating custom menus. You can choose to add an application that is already in the GNOME Applications menu or to create an application launcher for one that is not. Launchers can be added to a panel by dragging them directly. Launchers include applications, windows, and files.

Application Launchers

If an application already has an application launcher, it's easy to add it to a panel. You just drag the application launcher to the panel. This will automatically create a copy of the launcher for use on that panel. Launchers can be menu items or desktop icons. All the



Figure 8-7
Add to Panel Box listing panel objects

entries in your Applications menu are application launchers. To add an application from the menu, just select it and drag it to the panel. You can also drag any desktop application icon to a panel to add a copy of it to that panel.

For any menu item, you can also go to its entry and right-click it. Then select the Add This Launcher To Panel entry. An application launcher for that application is then automatically added to the panel. Suppose you use gedit frequently and want to add its icon to the panel, instead of having to go through the Applications menu all the time. Right-click the Text Editor menu entry in the Accessories menu, and select the Add This Launcher To Panel option. The gedit icon now appears in your panel.

You can also select the Add To Panel entry from the panel menu and then choose the Application Launcher entry. This will display a box with a listing of all the Applications menu entries along with Preferences and Administration menus, expandable to their items. Just find the application you want added and select it. This may be an easier approach if you are working with many different panels.

Keep in mind that for any launcher that you previously created on the desktop, you can just drag it to the panel to have a copy of the launcher placed on the panel.

Folder and File Launchers

To add a folder to a panel, just drag it directly from the file manager window or from the desktop. To add a file, you can also drag it to directly to the panel, but you will then have to create a launcher for it. The Create Launcher window will be displayed, and you can give the file launcher a name and select an icon for it.

Adding Drawers

You can also group applications under a Drawer icon. Clicking the Drawer icon displays a list of the different application icons you can then select. To add a drawer to your panel, right-click the panel and select the Add To Panel entry to display the Add To list. From that list select the Drawer entry. This will create a drawer on your panel. You can then drag any items from the desktop, menus, or windows to the drawer icon on the panel to have them listed in the drawer.

If you want to add, as a drawer, a whole menu of applications on the main menu to your panel, right-click any item in that menu, select Entire Menu from the pop-up menu, and then select the Add This As Drawer To Panel entry. The entire menu appears as a drawer on your panel, holding icons instead of menu entries. For example, suppose you want to place the Internet Applications menu on your panel. Right-click any entry item in the Internet Applications menu, select Entire Menu, and select Add This As Drawer To Panel. A drawer appears on your panel labeled Internet Applications, and clicking it displays a pop-up list of icons for all the Internet applications.

Adding Menus

A menu differs from a drawer in that a *drawer* holds application icons instead of menu entries. You can add menus to your panel much as you add drawers. To add a submenu from the Applications menu to your panel, right-click any item and select Entire Menu, and then select the Add This As Menu To Panel entry. The menu title appears in the panel; you can click it to display the menu entries.

In addition, you can add a menu from the panel's Add To list by selecting Custom Menu.

Adding Folders

You can also add directory folders to a panel. Click and drag the Folder icon from the file manager window to your panel. Whenever you click this Folder button, a file manager window opens, displaying that directory. You already have a Folder button for your home directory. You can add directory folders to any drawer on your panel.

Special Panel Objects

Special panel objects perform operations not supported by other panel objects. Currently, these include the Lock, Logout, and Launcher buttons, as well as the status dock. The Lock button, which displays a padlock, will lock your desktop, running the screensaver in its place. To access your desktop, click it and then enter your user password at the password prompt. The Logout button shows an open door. Clicking it will display the Logout dialog box, and you can then log out. It is the same as selecting Logout from the desktop menu. The Launcher button shows a launcher icon. It opens the Create Launcher dialog box, which allows you to enter or select an application to run.

The status dock is designed to hold status docklets. A status docklet provides current status information on an application. KDE applications that support status docklets can use the GNOME status dock when run under GNOME.

GNOME Applets

Applets are small programs that perform tasks within the panel. To add an applet, right-click the panel and select Add To Panel from the pop-up menu. This displays the Add To box, listing common applets along with other types of objects, such as launchers. Select the applet you want. For example, to add the clock to your panel, select Clock from the panel's Add To box. Once added, the applet will show up in the panel. If you want to remove an applet, right-click it and select the Remove From Panel entry.

GNOME features a number of helpful applets. Some applets monitor your system, such as the Battery Charge Monitor, which checks the battery in laptops, and System Monitor,

which shows a graph indicating your current CPU and memory use. The Volume Control applet displays a small scroll bar for adjusting sound levels. The Deskbar tool searches for files on your desktop. Network Monitor lets you monitor a network connection.

Several helpful utility applets provide added functionality to your desktop. The Clock applet can display time in a 12- or 24-hour format. Right-click the Clock applet and select the Preferences entry to change its setup. The CPU Frequency Scaling Monitor displays CPU usage for CPUs like AMD and the new Intel processors that run at lower speeds when idle.

Workspace Switcher

The *Workspace Switcher* appears in the panel and shows a view of your virtual desktops (see Figure 8-8). Virtual desktops are defined in the window manager. Located on the right side of the lower panel, the Workspace Switcher lets you easily move from one desktop to another with the click of a mouse. It is a panel applet that works only in the panel. You can add the Workspace Switcher to any panel by selecting it from that panel's Add To box.

The Workspace Switcher shows your entire virtual desktop as separate rectangles listed next to each other. Open windows show up as small colored rectangles in these squares. You can move any window from one virtual desktop to another by clicking and dragging its image in the Workspace Switcher. To configure the Workspace Switcher, right-click it and select Preferences to display the Preferences dialog box. Here, you can select the number of workspaces. The default is four.

GNOME Window List

The *Window List* shows currently opened windows (see Figure 8-8). The Window List arranges opened windows in a series of buttons, one for each window. A window can include applications such as a web browser, or it can be a file manager window displaying a directory. You can move from one window to another by clicking its button. When you minimize a window, you can later restore it by clicking its entry in the Window List.

Right-clicking a window's Window List button opens a menu that lets you Minimize or Unminimize, Roll Up, Move, Resize, Maximize or Unmaximize, or Close the window. The Minimize operation reduces the window to its Window List entry. Right-clicking the entry displays the menu with an Unminimize option instead of a Minimize one, which you can then use to redisplay the window. The Roll Up entry reduces the window to its title bar. The Close entry closes the window, ending its application.

If there is not enough space on the Window List applet to display a separate button for each window, then common windows will be grouped under a button that will expand like a menu, listing each window in that group. For example, all open terminal windows would be grouped under a single button, which when clicked would pop up a list of their buttons.

The Window List applet is represented by a small serrated bar at the beginning of the window button list. To configure the Window List, right-click this bar and select the Properties entry. Here, you can set features such as the size in pixels, whether to group windows, whether to show all open windows or those from just the current workspace, or which workspace to restore windows to.



Figure 8-8 Panel with Workspace Switcher and Window List, at the bottom of the desktop

GNOME Configuration

You can configure different parts of your GNOME interface using tools listed in the Preferences menu in the System menu. This menu will display entries for the primary GNOME preferences, organized into submenu categories like Hardware and Personal, along with preferences listing task-specific tools, like those for the Palm Pilot or Desktop Switcher. Selecting one will open a window labeled with the tool name, such as mouse preferences.

Your GNOME system provides several desktop tools you can use to configure your desktop, such as Desktop Background, Screensaver, and Themes. You use the Desktop Background applet to select a background color or image, the Screensaver to select the screen saver images and wait time, and the Theme tool to choose a theme (see Figure 8-9).

The Removable Drives and Media Preferences tools let you set what actions to perform on removable drives, CD and DVD discs, and digital cameras.

For sound configuration, the Sound tool lets you select sound files to play for events in different GNOME applications. For your keyboard, you can set the repeat sensitivity and click sound with the Keyboard tool. You can configure mouse buttons for your right or left hand and adjust the mouse motion.

GNOME Directories and Files

GNOME binaries are usually installed in the **/usr/bin** directory on your system. GNOME libraries are located in the **/usr/lib** directory. GNOME also has its own **include** directories with header files for use in compiling and developing GNOME applications, **/usr/include/libgnome-2.0/libgnome** and **/usr/include/libgnomeui** (see Table 8-5). These are installed by of the GNOME development packages. The directories located in **/usr/share/gnome** contain files used to configure your GNOME environment.

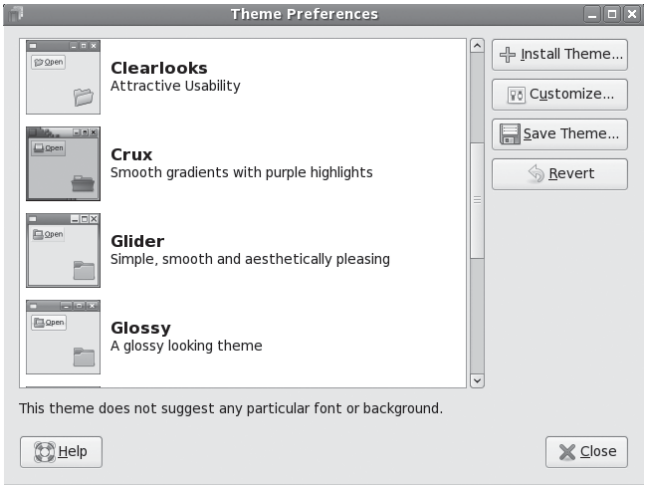


FIGURE 8-9 Selecting GNOME themes

GNOME System Directory	Contents
/usr/bin	GNOME programs
/usr/lib	GNOME libraries
/usr/include/libgnome-2.0/libgnome	Header files for use in compiling and developing GNOME applications
/usr/include/libgnomeui	Header files for use in compiling and developing GNOME user interface components
/usr/share/gnome	Files used by GNOME applications
/usr/share/doc/gnome*	Documentation for various GNOME packages, including libraries
/etc/gconf	GConf configuration files
GNOME User Directory	**Contents**
.gnome, .gnome2	Configuration files for the user's GNOME desktop and GNOME applications; includes configuration files for the panel, background, MIME types, and sessions
DESKTOP	Directory where files, directories, and links you place on the desktop will reside
.gnome2_private	The user's private GNOME directory
.gtkrc	GTK+ configuration file
.gconf	GConf configuration database
.gconfd	GConf **gconfd** daemon management files
.gstreamer	GNOME GStreamer multimedia configuration files
.nautilus	Configuration files for the Nautilus file manager

Table 8-5 GNOME Configuration Directories

GNOME User Directories

GNOME sets up several configuration files and directories in your home directory. The **.gnome**, **.gnome2,** and **.gconf** directories hold configuration files for different desktop components, such as **nautilus** for the file manager and **panel** for the panels. The **DESKTOP** directory holds all the items you placed on your desktop. The **.gtckrc** file is the user configuration file for the GTK+ libraries, which contains current desktop configuration directives for resources such as key bindings, colors, and window styles.

The GConf Configuration Editor

GConf provides underlying configuration support (not installed by default). GConf corresponds to the Registry used on Windows systems. It consists of a series of libraries used to implement a configuration database for a GNOME desktop. This standardized configuration database allows for consistent interactions between GNOME applications. GNOME applications that are built from a variety of other programs, as Nautilus is, can

use GConf to configure all those programs according to a single standard, maintaining configurations in a single database. Currently the GConf database is implemented as XML files in the user's **.gconf** directory. Database interaction and access is carried out by the GConf daemon **gconfd**.

You can use the GConf editor to configure different GNOME applications and desktop functions. To start the GConf editor, enter **`gconf-editor`** in a terminal window, or select Configuration Editor from the Applications | System Tools menu (Applications menu). Be sure to install the gconf-editor package first (you can use Pirut—Add/Remove Software).

Configuration elements are specified keys that are organized by application and program. You can edit the keys, changing their values. Figure 8-10 shows the GConf editor settings for the dialog display features used for the GNOME interface.

The GConf editor has four panes:

- **Tree** The pane for navigating keys, with expandable trees for each application, is located on the left. Application entries expand to subentries, grouping keys into different parts or functions for the application.
- **Modification** The pane at the top right will display the keys for a selected entry. The name field will include an icon indicating its type, and the Value field is an editable field showing the current value. You can directly change this value.
- **Documentation** The pane at the bottom right displays information about the selected key, showing the key name, the application that owns it, a short description and a detailed description.
- **Results** This pane, displayed at the bottom, only appears when you do a search for a key.

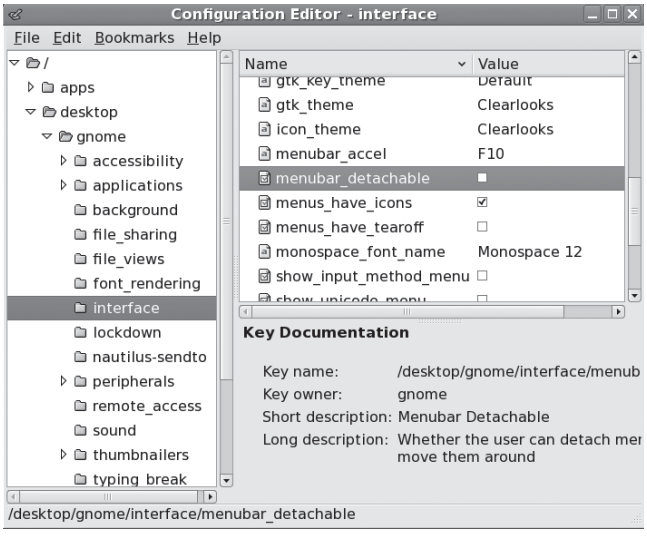


FIGURE 8-10
GConf editor

A key has a specific type, such as numeric or string, and you will only be able to make changes using the appropriate type. Each key entry has an icon specifying its type, such as a check mark for the Boolean values, a number *1* for numeric values, and a letter a for string values. Some keys have pop-up menus with limited selections to choose from, represented by an icon with a row of lines. To change the value of a key, click its value field. You can then edit the value. For pop-up menus, you right-click the value field to display the menu.

There are many keys distributed over several applications and groups. To locate one, you can use the search function. Select Find from the Edit menu and enter a pattern. The results are displayed in a Results pane, which you can use to scroll through matching keys, selecting the one you want.

Changes can be made either by users or by administrators. Administrators can set default or mandatory values for keys. Mandatory values will prevent users from making changes. For user changes, you can open a Settings window by selecting Settings from the File menu. This opens an identical GConf editor window. For administrative changes, you first log in as the root user. For default changes, you select the Default entry from the File menu, and for mandatory changes, select the Mandatory entry.

CHAPTER 9

KDE

The *K Desktop Environment (KDE)* is a network-transparent desktop that includes the standard desktop features, such as a window manager and a file manager, as well as an extensive set of applications that covers most Linux tasks. KDE is an Internet-aware system that includes a full set of integrated network/Internet applications, including a mailer, a newsreader, and a web browser. The file manager doubles as a Web and FTP client, enabling you to access Internet sites directly from your desktop. KDE aims to provide a level of desktop functionality and ease of use found in Macintosh and Windows systems, combined with the power and flexibility of the Unix operating system.

The KDE desktop is developed and distributed by the KDE Project, which is a large open group of hundreds of programmers around the world. KDE is entirely free and open software provided under a GNU Public License and is available free of charge along with its source code. KDE development is managed by a core group: the KDE Core Team. Anyone can apply, though membership is based on merit.

NOTE *KDE applications are developed using several supporting KDE technologies, including KIO, which offers seamless and modular access to files and directories across a network. For interprocess communication, KDE uses the Desktop Communications Protocol (DCOP). KParts is the KDE component object model used to embed an application within another, such as a spreadsheet within a word processor. KHTML is an HTML rendering and drawing engine.*

Numerous applications written specifically for KDE are easily accessible from the desktop. These include editors, photo and paint image applications, spreadsheets, and office applications. Such applications usually have the letter *K* as part of their name—for example, KWord or KMail. A variety of tools are provided with the KDE desktop. These include calculators, console windows, notepads, and even software package managers. On a system administration level, KDE provides several tools for configuring your system. With KUser, you can manage user accounts, adding new ones or removing old ones. Practically all your Linux tasks can be performed from the KDE desktop. KDE applications also feature a built-in Help application. Choosing the Contents entry in the Help menu starts the KDE Help viewer, which provides a web page–like interface with links for navigating through the Help documents. KDE version 3 includes support for the office application suite KOffice, based on KDE's KParts technology. KOffice includes a presentation application, a spreadsheet, an illustrator, and a word processor, among other components. In addition, an integrated

Website	Description
kde.org	KDE website
ftp.kde.org	KDE FTP site
kde-apps.org	KDE software repository
developer.kde.org	KDE developer site
trolltech.com	Site for Qt libraries
Koffice.org	KOffice office suite
kde-look.org	KDE desktop themes, select KDE entry
lists.kde.org	KDE mailing lists

TABLE 9-1 KDE Websites

development environment (IDE), called KDevelop, is available to help programmers create KDE-based software.

NOTE *On KDE, menus will show more KDE applications than are shown on GNOME, including access to the KDE Control Center on the main menu.*

KDE, which was initiated by Matthias Ettrich in October 1996, has an extensive list of sponsors, including SuSE, Red Hat, Fedora, Mandrake, O'Reilly, and others. KDE is designed to run on any Unix implementation, including Linux, Solaris, HP-UX, and FreeBSD. The official KDE website is **kde.org**, which provides news updates, download links, and documentation. KDE software packages can be downloaded from the KDE FTP site at **ftp.kde.org** and its mirror sites. Several KDE mailing lists are available for users and developers, including announcements, administration, and other topics (see the KDE website to subscribe). A great many software applications are currently available for KDE at **kde-apps.org**. Development support and documentation can be obtained at **developer.kde.org**. Various KDE websites are listed in Table 9-1.

NOTE *Currently, new versions of KDE are being released frequently, sometimes every few months. KDE releases are designed to enable users to upgrade their older versions easily. The distribution updater should automatically update KDE from distribution repositories, as updates become available. Alternatively, you can download new KDE packages from your distribution's FTP site and install them manually. Packages tailored for various distributions can also be downloaded through the KDE website at **kde.org** or directly from the KDE FTP site at **ftp.kde.org** and its mirror sites in the **stable** directory.*

The Qt Library

KDE uses as its library of GUI tools the Qt library, developed and supported by Trolltech (**trolltech.com**). Qt is considered one of the best GUI libraries available for Unix/Linux systems. Using Qt has the advantage of relying on a commercially developed and supported GUI library. Also, using the Qt libraries drastically reduces the development time for KDE.

Trolltech provides the Qt libraries as open source software that is freely distributable. Certain restrictions exist, however: Qt-based (KDE) applications must be free and open-sourced, with no modifications made to the Qt libraries. If you develop an application with the Qt libraries and want to sell it, then you have to buy a license from Trolltech. In other words, the Qt library is free for free and open source applications, but not for commercial ones.

Configuration and Administration Access with KDE

KDE uses a different set of menus and access points than GNOME for accessing system administration tools. There are also different ways to access KDE configuration tasks, as well as KDE system administration tools not available through GNOME.

- **Control Center** Accessible from the main menu, Control Center, or from any file manager window's Go menu | Settings entry. This is the comprehensive KDE configuration tool that lists all the KDE configuration panels for managing your desktop, file manager, and system, as well as KDE's own administration tools that can be used instead of the GNOME ones.
- **System** Accessible from the main menu entry System. This is a collection of system tools. Here you will also find KDE administration tools, such as KUser for managing users.
- **Settings** Accessible from the main menu entry Settings. This is a smaller collection of desktop configuration features, such as setting your login photo, configuring the panel, and configuring printers.
- **Utilities** Accessible from the main menu entry Utilities. Here you will find tools for specific tasks like KPilot for Palm handhelds (under Peripherals) and Beagle searching.

One point of confusion is that Settings is used differently in the main menu and in the File Manager Go menu. In the main menu it refers to an ad hoc collection of desktop configuration tools such as mail notification, whereas in the File Manager window Go menu, it displays the Control Center, but in window icon format.

The term *System* is also used differently. In the file manager it displays desktop resources such as your Home directory or network shares, whereas in the main menu it lists system tools like the software updater. Also, on the panel you can add a System applet, which also lists the file manager system resources.

To initially configure your desktop, you may want to run the Desktop Configuration Wizard, accessible from main menu Settings. Here you can set your country and language, your desktop appearance, eye-candy feature level, and the desktop theme you want.

Finally, to configure only your desktop, you can right-click it anywhere to display a pop-up menu that will have a Desktop Configuration entry. Selecting this opens the Desktop Configuration panels. These are the same that are used on Control Center's Desktop selection.

The KDE Desktop

One of KDE's aims is to provide users with a consistent integrated desktop, where all applications use GUIs (see Figure 9-1). To this end, KDE provides its own window manager (KWM), file manager (Konqueror), program manager, and desktop panel (Kicker). You can



FIGURE 9-1 The KDE desktop

run any other X Window System–compliant application, such as Firefox, in KDE, as well as any GNOME application. In turn, you can also run any KDE application, including the Konqueror file manager, in GNOME.

NOTE *When you start KDE for the first time, you will be prompted to configure the Knemo network device monitor. Here you can specify your network devices like eth0 for the first Ethernet device, ppp0 for dialup connection, or wlan0 for wireless connections. You can also specify tooltips to use, the icon for monitoring, even the color.*

KDE Menus

When you start KDE, the KDE panel is displayed at the bottom of the screen. Located on the panel are icons for menus and programs, as well as buttons for different desktop screens. The button for the main menu shows a *k*, the KDE icon. This is the button for the KDE main menu. Click this button to display the menu of applications you run (you can also open the main menu by pressing ALT-F1). From the KDE menu, you can access numerous submenus for different kinds of applications. The menu also includes certain key items such as Logout, to log out of KDE; Lock Session, to lock your desktop; Control Center, to configure your KDE desktop; Switch User, to log in to another user without logging out first; Run Command, to run programs from a command line; Home directory, to quickly browse your home directory; and Help, which starts the KDE help tool.

TIP *You can run the Desktop Settings Wizard on the Settings menu to easily change your desktop settings.*

The standard KDE applications installed with the KDE can be accessed through this menu. The main menu has most of the same entries as those found on GNOME. The entries have been standardized for both interfaces. You can find entries for categories such as Internet, System Settings, Graphics, and Office. These menus list both GNOME and KDE applications you can use. However, some of the KDE menus contain entries for a few more alternate KDE applications, like KMail on the Internet menu. Some entries will invoke the KDE version of a tool, like the Terminal entry in the System Tools menu, which will invoke the KDE terminal window, Konsole. In addition, the Preferences menu is nearly empty, with only the same More Preferences submenu. In GNOME, the Preferences menu is used specifically to configure GNOME. To configure KDE, you use the KDE Control Center referenced by the Control Center item in the main menu.

TIP *If your CD or DVD-ROM device icons are not displayed when you insert a CD/DVD, you will need to enable device icon display on your desktop. Right-click the desktop and choose Configure Desktop from the pop-up menu. This shows only the desktop entries of the Control Center. Select Behavior, and then on the Device Icons pane, select the Show Device Icons check box. A long list of connectable devices is displayed, with default devices already selected. You select and deselect the ones you want shown or hidden. For most devices, you have both mounted and unmounted options. For example, an unmounted entry for the DVD-ROM will display a DVD-ROM icon even if the DVD-ROM device is empty.*

Quitting KDE

To quit KDE, you can either select the Logout entry in the main menu or right-click anywhere on the desktop and select the Logout entry from the pop-up menu. If you leave any KDE or X11 applications or windows open when you quit, they are automatically restored when you start up again. If you just want to lock your desktop, you can select the Lock Screen entry on the main menu and your screen saver will appear. To access a locked desktop, click the screen and a box appears prompting you for your login password. When you enter the password, your desktop reappears.

NOTE *You can use the Create New menus to create new folders or files on the desktop, as well as links for applications and devices.*

KDE Desktop Operations

Initially the Trash icon is shown on the left side. The Trash icon operates like the Recycle Bin in Windows or the trash can on the Mac. Drag items to it to hold them for deletion. You can use the floppy and DVD/CD-ROM icons to mount, unmount, and display the contents of CD-ROMs and floppy disks. The DVD/CD-ROM icons will appear as discs are inserted and mounted, disappearing when ejected. Your home directory is accessed initially from the Home directory in the main menu or an icon in the panel. To place this folder on the desktop, right-click the Home entry in the main menu and select Add to Desktop from the pop-up menu. A home directory icon then appears permanently on your desktop. You also have the option to display the Home directory on your panel by selecting Add to Panel.

The KDE panel displayed across the bottom of the screen initially shows small buttons for the KDE main menu, the user's home directory, the web browser, office tools, a clock, and buttons for virtual desktops, among others. The desktop supports drag-and-drop operations. For example, to print a document, drag it to the Printer icon. You can place any directories on the desktop by simply dragging them from a file manager window to the desktop. A small menu will appear with options to copy or link the folder. To just create an icon on the desktop for the same folder, select the link entry.

The desktop also supports copy-and-paste operations, holding text you copied from one application in a desktop clipboard that you can then use to paste into another application. You can even copy and paste from a Konsole window. For example, you can copy a web address from a web page and then paste it into an email message or a word processing document. This feature is supported by the Klipper utility located on the panel.

You can create new directories on the desktop by right-clicking anywhere on the desktop and selecting Create New and then Directory from the pop-up menu. All items that appear on the desktop are located in the **Desktop** directory in your home directory. There you can find the **Trash** directory, along with any others you place on the desktop. You can also create simple text files and HTML files using the same menu.

Accessing System Resources from the File Manager

You can access system resources like network shares, user directories, or storage media like CD-ROMs, from any file manager window (see Figure 9-2). Some of these resources can be opened directly, like the Trash and CD-ROM icons on your desktop or the Home directory entry in the main menu. To open an initial file manager window, select the Home entry in the main menu. Then click the system icon on the lower left side of the file manager window (last icon). Icons are displayed that you can use to access various system resources, such as storage media, remote hosts, Samba network shares, and your trash folder. You will see icons labeled Home Folder, Remote Places, Storage Media, and Trash. You can also access any of these directly from the file manager Go menu. The Storage Media icon will expand to list all your DVD/CD media, removable media, and hard disk partitions. Under Remote Places you will find icons for your local network, network services, Samba shares, and a tool to add network folders. Through the Samba Shares icon you can access your shared Windows folders and printers. The Storage Media icon lists your storage media such as your CD-ROMs. You can open these to access their contents. Certain resources have their

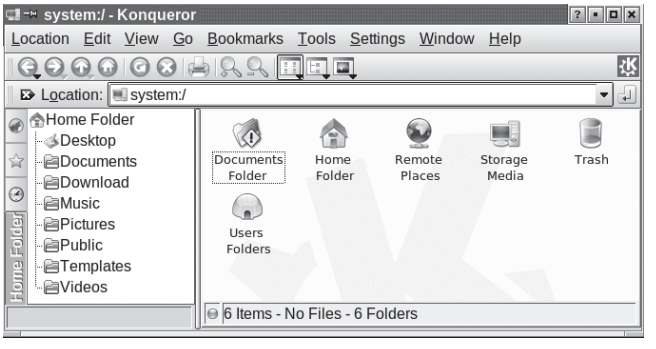


FIGURE 9-2 System resources access from the file manager

own URLs that you can enter into a file manager location box to directly access them; for example, Remote Places has the URL **remote:/**, Samba uses **smb:/**, and Storage Media uses **media:/**.

Configuring Your Desktop

To configure your desktop, right-click the desktop and select the Configure Desktop entry. This displays a window with entries for Behavior, Multiple Desktops, Display, Background, and Screensaver. All these features can be configured also using the KDE Control Center's Appearance & Themes panels:

- Display lets you set the display resolution and orientation.
- Behavior lets you enable the display of certain features, such as displaying a desktop menu across the top of the screen, or showing icons on the desktop. You can also select the operations for a mouse click on the desktop. The right-click currently displays the desktop menu. You can also specify which devices to display on the desktop.
- The Multiple Desktop panel lets you select the number of virtual desktops to display.
- Background lets you choose a background color or image for each virtual desktop.
- Screensaver lets you select a screen saver along with its timing. Numerous screen savers are already configured.

For your desktop, you can also select a variety of different themes. A *theme* changes the look and feel of your desktop, affecting the appearance of GUI elements, such as scroll bars, buttons, and icons. For example, you use the Mac OS theme to make your K Desktop look like a Macintosh. You can use the Theme Manager in the KDE Control Center (Appearance & Themes) to select a theme and install new ones. Several may be installed for you, including Bluecurve (Fedora) or the Default (the KDE theme). Additional themes for the K Desktop can be downloaded from the **kde-look.org** website.

Desktop Link Files and URL Locations

On the KDE desktop, special files called *link* files are used to access a variety of elements, including websites, application programs, and even devices. You create a link file by right-clicking the desktop and then selecting Create New. From this menu, you choose the type of link file you want to create.

The Link To Application entry is for launching applications. The Link To Location (URL) entry holds a URL address that you can use to access a website or FTP site. The Link to Device submenu lets you create links to different kinds of devices, including CD-ROMs, hard disks, and cameras. Bear in mind that these are links only. You rarely need to use them. Device icons that display on your desktop are now automatically generated directly by udev and HAL as needed.

To create a URL desktop file, right-click the desktop and select the Create New menu and the File submenu. Then select the Link To Location (URL) entry. A window appears that displays a box that prompts you to enter a name for the file and the URL address. Be sure to precede the URL with the appropriate protocol, like **http://** for web pages. Alternatively, you

can simply drag and drop a URL directly from the Location box on a web browser such as Firefox. You can later edit the desktop file by right-clicking it and selecting Properties. A desktop dialog box for URL access is then displayed. This dialog box has three tabbed panels: General, Permissions, and URL. On the General panel is the name of your desktop file. It will have as its name the name that you entered. An Icon button on this panel shows the icon that will be displayed for this desktop file on your desktop. You can select an icon by clicking the Icon button to open a window that lists icons you can choose from. Click OK when you are finished. The desktop file then appears on your desktop with that icon. On the URL panel, you will see a box labeled URL with a URL you entered already in it. You can change it if you want. For example, for online themes, the URL would be **http://www.kde-look.org**.

On your desktop, you can click the URL icon anytime to access that website. An alternative and easier way to create a URL desktop file is simply to drag a URL for a web page displayed on the file manager to your desktop. A pop-up window will let you select Copy or Link. Choose Link to create a URL desktop file (Copy will create local copy of that page). A desktop file is automatically generated with that URL. To change the default icon used, you can right-click the file and choose Properties to display the desktop dialog box.

KDE Windows

A KDE window has the same functionality you find in other window managers and desktops. You can resize the window by clicking and dragging any of its corners or sides. A click-and-drag operation on a side extends the window in that dimension, whereas on a corner it extends both height and width at the same time. Notice that the corners are slightly enhanced. The top of the window has a title bar showing the name of the window, the program name in the case of applications, and the current directory name for the file manager windows. The active window has the title bar highlighted. To move the window, click this title bar and drag it where you want. Right-clicking the window title bar displays a drop-down menu with entries for window operations, such as closing or resizing the window. Within the window, menus, icons, and toolbars for the particular application are displayed.

You can configure the appearance and operation of a window by selecting the Configure Window Behavior entry from the Window menu (right-click the title bar). Here you can set appearance (Window Decorations); button and key operations (Actions); the focus policy, such as a mouse click on the window or just passing the mouse over it (Focus), how the window is displayed when moving it (Moving); and advanced features like moving a window directly to another virtual desktop (Active Desktop Borders). All these features can be configured also using the KDE Control Center's Appearance & Themes panels.

Opened windows are also shown as buttons on the KDE taskbar located on the panel. The taskbar shows the different programs you are running or windows you have open. This is essentially a docking mechanism that lets you change to a window or application just by clicking its button. When you minimize (iconify) a window, it is reduced to its taskbar button. You can then restore the window by clicking its taskbar button.

To the right of the title bar are three small buttons for minimizing, maximizing, or closing the window. You can switch to a window at any time by clicking its taskbar button. From the keyboard, you can use the ALT-TAB key combination to display a list of current applications. Holding down the ALT key and sequentially pressing TAB moves you through the list.

Application windows may display a Help Notes button, shown next to the iconify button and displaying a question mark. Clicking this button changes your cursor to a question mark.

You can then move the cursor to an item such as an icon on a toolbar and click it to display a small help note explaining what the item does. For example, moving the mouse to the Forward button in the file manager taskbar will show a note explaining that this button performs a browser forward operation.

TIP *The taskbar and pager has three styles: elegant, classic, and transparent.*

Virtual Desktops: The KDE Desktop Pager

KDE, like most Linux window managers, supports virtual desktops. In effect, this extends the desktop area on which you can work. You could have Mozilla running on one desktop and use a text editor in another. KDE can support up to 16 virtual desktops, though the default is 4. Your virtual desktops can be displayed and accessed using the KDE Desktop Pager located on the panel. The KDE Desktop Pager represents your virtual desktops as miniature screens showing small squares for each desktop. It is made to look similar to the GNOME Workspace Switcher. The default four squares are numbered 1, 2, 3, and 4. To move from one desktop to another, click the square for the destination desktop. Clicking 3 displays the third desktop, and clicking 1 moves you back to the first desktop. If you want to move a window to a different desktop, first open the window's menu by right-clicking the window's title bar. Then select the To Desktop entry, which lists the available desktops. Choose the one you want.

You can also configure KDE so that if you move the mouse over the edge of a desktop screen, it automatically moves to the adjoining desktop. You need to imagine the desktops arranged in a four-square configuration, with two top desktops next to each other and two desktops below them. You enable this feature by enabling the Active Desktop Borders feature in the Desktop | Window Behavior | Advanced panel in the KDE Control Center.

To change the number of virtual desktops, use the KDE Control Center's Desktop entry. Either select the Configure Desktop entry in the desktop pop-up menu (right-click anywhere on the desktop background) and choose Multiple Desktops, or select Control Center from the main menu and open the Desktop heading to select the Multiple Desktops entry. The visible bar controls the number of desktops. Slide this to the right to add more and to the left to reduce the number. You can change any of the desktop names by clicking a name and entering a new one. In the Appearance & Themes' Background entry, you can change the appearance for particular desktops such as color background and wallpaper (deselect Common background first).

TIP *Use the CTRL-TAB keys to move to the next desktop, and CTRL-SHIFT-TAB to go the previous desktop. Use the CTRL key in combination with a function key to switch to a specific desktop, for example, CTRL-F1 switches to the first desktop and CTRL-F3 to the third desktop.*

KDE Panel: Kicker

The KDE panel (Kicker), located at the bottom of the screen, provides access to most KDE functions (see Figure 9-3). The panel includes icons for menus, directory windows, specific programs, and virtual desktops. At the left end of the panel is a button for the main menu (also known as the K menu), a KDE *K* icon.



FIGURE 9-3 KDE panel

To add an application to the panel, right-click anywhere on the panel and select Add from the pop-up menu. The Add menu displays the kind of objects you can add, including applets, applications, panel extensions, and special buttons. For KDE applications, select the Applications entry. This lists all installed KDE applications on your main menu. Click the application name to add an application button to the panel. You can also drag applications from a file manager window or from the main menu to the panel directly and have them automatically placed in the panel. The panel displays only desktop files. When you drag and drop a file to the panel, a desktop file for it is automatically generated.

Kicker also support numerous applets and several panel extensions, as well as special buttons.

- Applets are designed to run as icons in the panel. These include a clock, a pager, and a system monitor.
- Panel extensions add components to your desktop (select Panel from the Add menu). For example, the Kasbar extension sets up its own panel and lists icons for each window you open. You can easily move from one window to another by clicking the corresponding icon in the Kasbar extension panel.
- Special buttons, including buttons for KDE-specific operations like the KDE Window list, a Kterm terminal window, the KDE print manager, and KDE preferences.

To configure the panel position and behavior, right-click the panel and select the Configure Panel entry. This displays a customized control module window that collects the Panel configuration entries from the KDE Control Center. There are five configuration windows. The first four let you determine how the panel is displayed, and the last, Taskbar, configures how windows are shown on the taskbar. These conform to the KDE Control Center's Desktop | Panels | Taskbar entries.

The first four panes are Arrangement, Hiding, Menus, and Appearance. The Arrangement pane enables you to specify the edges of the screen where you want your panel and taskbar displayed. You can also enlarge or reduce it in size. The Hiding pane lets you set the hiding mode, whether to enable auto-hiding or to manually hide and display the taskbar. The Menus pane lets you control the size of your menus as well as whether to display recently opened documents as menu items. You can also select certain default entries like Preferences and Bookmarks, as well as edit the K menu directly, adding or removing items. The Appearance pane lets you set button colors for buttons and the background image for the taskbar. With the Taskbar pane, you can control windows and tasks displayed on the taskbar, as well as set the button actions.

The KDE Help Center

The KDE Help Center provides a browserlike interface for accessing and displaying both KDE Help files and Linux Man and info files. You can start the Help Center by selecting its entry in the main menu (the life preserver), or by right-clicking the desktop and selecting the Help entry. The Help window is divided into two frames. The left frame of the Help screen holds two tabbed panels, one listing contents and the other providing a glossary. The right frame displays currently selected documents. A help tree on the contents panel lets you choose the kind of Help documents you want to access. Here you can choose manuals,

Man pages, or info documents, even application manuals. The Help Center includes a detailed user manual, a FAQ, and KDE website access.

A navigation toolbar enables you to move through previously viewed documents. KDE Help documents use an HTML format with links you can click to access other documents. The Back and Forward commands move you through the list of previously viewed documents. The KDE Help system provides an effective search tool for searching for patterns in Help documents, including Man and info pages. Select the Find entry from the Edit menu to display a page where you can enter your pattern.

Applications

You can start an application in KDE in several ways. If an entry for it is in the main menu, select that entry to start the application. Some applications also have buttons on the KDE panel you can click to start them. Depending on the distribution, the panel will initially hold applications like the Firefox web browser and several Office.org applications. You can also use the file manager to locate a file that uses that application. Clicking the file's icon starts the application. Another way to start an application is to open a shell window, enter the name of the application at the shell prompt, and press ENTER. You can also select Run Command from the main menu (or press ALT-F2) to open a small window consisting of a box to enter a single command. Previous commands can be accessed from a pop-up menu. An Options button will list options for running the program, such as priority or within a terminal window.

NOTE *You can create a desktop file on your desktop for any application already on your KDE menu by simply clicking and dragging its menu entry to the desktop. Select Copy, and a desktop file for that application is created for you on your desktop, showing its icon.*

To create a new desktop file for an application, right-click anywhere on the empty desktop, select Create New from the pop-up menu, and then within the File submenu, choose Link To Application. Enter the name for the program and a desktop file for it appears on the desktop with that name. A Properties dialog box then opens with four panels: General, Permissions, Application, and Preview. The General panel displays the name of the link. To select an icon image for the desktop file, click the icon. The Select Icon window is displayed, listing icons from which you can choose.

On the Permissions panel, be sure to set execute permissions so that the program can be run. You can set permissions for yourself, your group, or any user on the system. The Meta Info panel will list the type of file system used.

To specify the application the desktop file runs, go to the Application panel and either enter the application's program name in the Command box or click Browse to select it. On this panel, you also specify the description and comment. For the description, enter the application name. This is the name used for the link, if you use the file manager to display it. The comment is the Help note that appears when you pass your mouse over the icon.

In the Application panel, you can also specify the type of documents to be associated with this application. The bottom of the panel shows Add and Remove buttons. To specify a MIME type, click Add. This displays a list of file types and their descriptions. Select the one you want associated with this program. Desktop files needn't reside on the desktop. You can

place them in any directory and access them through the file manager. You can later make changes to a desktop file by right-clicking its icon and selecting Properties from the pop-up menu. This again displays the dialog box for this file. You can change its icon and even the application it runs.

The Advanced Options button contains execute options for the application, such as running it in a shell window or as a certain user. To run a shell-based program such as Vi, select the Run In Terminal check box and specify any terminal options. Startup options let you list the program in the system tray.

TIP *You can have KDE automatically display selected directories or start certain applications whenever it starts up. To do so, place links for these windows and applications in the* ***AutoStart*** *directory located in your* ***.kde*** *directory.*

Mounting Devices from the Desktop

To access a CD-ROM, place the CD-ROM into your CD-ROM drive and double-click the CD-ROM icon. The file manager window then opens, displaying the contents of the CD-ROM's top-level directory. To eject the CD, right-click the CD-ROM's icon and select Eject from the pop-up menu (you can also elect to just unmount the CD-ROM).

To access USB drives, just insert the USB drive into any USB port. The drive will be automatically detected and a file manager window will open, showing the contents of the drive. You can read, copy, move, and delete files on the USB drive. A USB drive icon will appear on the desktop. Moving the cursor over the icon displays detailed information about the drive, such as where it is mounted and how much memory is used. Right-clicking and selecting Properties will display General, Permissions, Meta Info (space used), and Mounting information. The USB drive menu also has an entry for transferring an image file to the digiKam tool (Download Photos with digiKam). To remove a USB drive, first right-click on the USB icon and select the Safely Remove entry. The USB drive icon will disappear from the desktop. You can then remove the drive.

To access a floppy disk, place the floppy disk into the disk drive and double-click the Floppy Disk icon. This displays a file manager window with the contents of the floppy disk. Be careful not to remove the disk unless you first unmount it. To unmount the disk, right-click its icon and select Unmount from the icon's pop-up menu. You can perform one added operation with floppy disks. If you put in a blank disk, you can format it. You can choose from several file system formats, including MS-DOS. To format a standard Linux file system, select the ext3 entry.

TIP *Never remove a USB drive directly, as you do with Windows. If you do so, any changes you made, like adding files, will not be done. First right-click on the USB drive icon and select Safely Remove. The USB drive icon will disappear from the desktop and you can then remove the USB drive.*

KDE File Manager and Internet Client: Konqueror

The KDE file manager, known as Konqueror, is a multifunctional utility with which you can manage files, start programs, browse the web, and download files from remote sites (see Figure 9-4). Traditionally, the term "file manager" was used to refer to managing files on

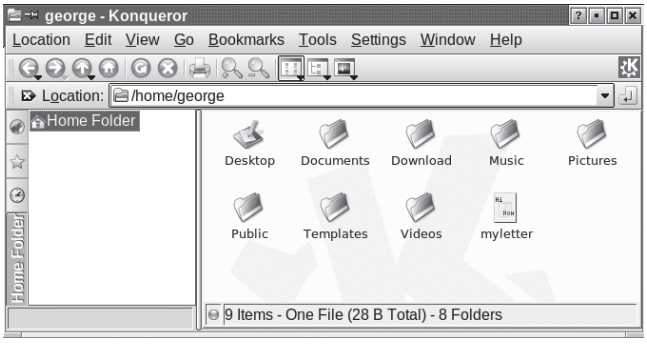


FIGURE 9-4
The KDE file manager

a local hard disk. The KDE file manager extends its functionality well beyond this traditional function because it is Internet capable, seamlessly displaying remote file systems as if they were your own, as well as viewing web pages with browser capabilities. It is capable of displaying a multitude of different kinds of files, including image, PostScript, and text files. KOffice applications can be run within the Konqueror window. You can even open a separate pane within a file manager window to run a terminal window where you can enter shell commands (the Window menu).

Konqueror Window

A KDE file manager window consists of a menu bar, a navigation toolbar, a location field, a status bar, and a sidebar that provides different views of user resources such as a tree view of file and directory icons for your home directory. When you first display the file manager window, it displays the file and subdirectory icons for your home directory. Files and directories are automatically refreshed. Thus if you add or remove directories, you do not have to manually refresh the file manager window. It automatically updates your listing, showing added files or eliminating deleted ones. The files listed in a directory can be viewed in several different ways, such as icons, multicolumn (small icons), expandable trees, file information, or in a detailed listing. The different views are listed in the View Mode submenu within the View menu, and the commonly used ones are listed as icons at the end of the icon bar. The Tree mode lists your subdirectories as expandable trees whose contents you can display by clicking their plus signs. The Info mode lists file information like the number of lines and characters in the file. The detailed listing provides permissions, owner, group, and size information. The Text view does the same but does not display an icon next to the filename.

Konqueror also supports tabbed displays. Instead of opening a folder in the same file manager window or a new one, you can open a new tab for it using the same file manager window. One tab can display the initial folder opened, and other tabs can be used for folders opened later. You can then move from viewing one folder to another by simply clicking the latter folder's tab. This way you can view multiple folders with just one file manager window. To open a folder as a tab, right-click its icon and select Open in New Tab. To later close the folder, right-click its tab label and select Close Tab. You can also detach a tab, opening it up in its own file manager window.

TIP *Configuration files, known as hidden files, are not usually displayed. To have the file manager display these files, select Show Hidden Files from the View menu. Konqueror also supports split views, letting you view different directories in the same window (the Window menu). You can split vertically or horizontally.*

You can open a file either by clicking it or by selecting it and then choosing the Open entry in the File menu. If you want to select the file or directory, you need to either hold down the CTRL key while you click it or single-click because a double-click opens the file. If the file is a program, that program starts up. If it is a data file, such as a text file, the associated application is run using that data file. For example, if you double-click a text file, the Kate application starts and displays that file. If Konqueror cannot determine the application to use, it opens a dialog box prompting you to enter the application name. You can click the Browse button on this box to use a directory tree to locate the application program you want.

The file manager can also extract tar archives. An *archive* is a file ending in **.tar.gz**, **.tar**, or **.tgz**. Clicking the archive lists the files in it. You can extract a particular file simply by dragging it out of the window. Clicking a text file in the archive displays it with Kate, while clicking an image file displays it with KView. For distributions supporting software packages like RPM and DEB, selecting the package opens it with the distribution's software install utility, which you can then use to install the package.

If the folder is a CVS folder, used for managing different versions of a project, you can use the Cervisia tool listed in the View Mode submenu to display and examine CVS archives.

Navigation Panel

The Navigation panel is a sidebar that lists different resources that a user can access with Konqueror. You can turn the Navigation panel on or off by selecting its entry in the Window menu. The sidebar is configured with the Navigation Panel Configuration tool, accessible as the first button on the Navigation panel's button bar.

TIP *Konqueror also provides a sidebar media player for running selected media files within your file manager window.*

The Navigation panel features a vertical button bar for displaying items, such as your bookmarks, devices, home directory, services, and network resources, in an expandable tree. Dragging the mouse over the resource icon displays its full name. When you click an item, its icon will expand to the name of that resource. Double-click it to access it with Konqueror. For example, to move to a subdirectory, expand your home directory entry and then double-click the subdirectory you want. Konqueror will now display that subdirectory. To go to a previously bookmarked directory or web page, find its entry in the Bookmarks listing and select it. The network button lists network resources you have access to, such as FTP and websites. The root folder button displays your system's root directory and its subdirectories.

To configure the Navigation panel, click its Configure button in the sidebar button bar. Select the Multiple Views entry to allow the display of several resource listings at once, each in its separate subsidebar. You can also add a new resource listing, choosing from a bookmark, history, or directory type. A button will appear for the new listing. You can right-click the button to select a new icon for it or select a URL, either a directory pathname or a network address. To remove a button and its listing, right-click it and select the Remove entry.

TIP *If the multiple views feature is enabled in the Navigation Panel Configuration, you can display several of these resources at once, just by clicking the ones you want. If this feature is not enabled, the previous listing is replaced by the selected one. Turn off a display by clicking its button again.*

Search

To search for files, select the Find entry in the Tools menu. This opens a pane within the file manager window in which you can search for filenames using wildcard matching symbols, such as *. Click Find to run the search and Stop to stop it. The search results are displayed in a pane in the lower half of the file manager window. You can click a file and have it open with its appropriate application. Text files are displayed by the Kate text editor. Images are displayed by KView, and PostScript files by KGhostView. Applications are run. The search program also enables you to save your search results for later reference. You can even select files from the search and add them to an archive.

Navigating Directories

Within a file manager window, a double-click on a directory icon moves to that directory and displays its file and subdirectory icons. To move back up to the parent directory, you click the up arrow button located on the left end of the navigation toolbar. A double-click on a directory icon moves you down the directory tree, one directory at a time. By clicking the up arrow button, you move up the tree. To move directly to a specific directory, you can enter its pathname in the Location box located just above the pane that displays the file and directory icons. Like a web browser, the file manager remembers the previous directories it has displayed. You can use the back and forward arrow buttons to move through this list of prior directories. You can also use several keyboard shortcuts to perform such operations, as listed in Table 9-2.

Keys	Description
ALT-LEFT ARROW, ALT-RIGHT ARROW	Backward and forward in History
ALT-UP ARROW	One directory up
ENTER	Open a file/directory
ESC	Open a pop-up menu for the current file
LEFT/RIGHT/UP/DOWN ARROWS	Move among the icons
SPACEBAR	Select/unselect file
PAGE UP, PAGE DOWN	Scroll up/down fast
CTRL-C	Copy selected file to clipboard
CTRL-V	Paste files from clipboard to current directory
CTRL-S	Select files by pattern
CTRL-L	Open new location
CTRL-F	Find files
CTRL-W	Close window

TABLE 9-2 KDE File Manager Keyboard Shortcuts

If you know you want to access particular directories again, you can bookmark them, much as you do a web page. Just open the directory and select the Add Bookmarks entry in the Bookmarks menu. An entry for that directory is then placed in the file manager's Bookmark menu. To move to the directory again, select its entry in the Bookmark menu. To navigate from one directory to another, you can use the Location field or the directory tree. In the Location field, you can enter the pathname of a directory, if you know it, and press ENTER. The directory tree provides a tree listing all directories on your system and in your home directory. To display the directory tree, select the Tree View from the View menu's View Mode submenu, or click the Tree View icon in the icon bar. To see the Tree View for your home or root directory directly, you can use the Navigation panel's Home or Root Folder resources.

Copy, Move, Delete, Rename, and Link Operations

To perform an operation on a file or directory, you first have to select it. To select a file or directory, you click the file's icon or listing. To select more than one file, continue to hold the CTRL key down while you click the files you want. You can also use the keyboard arrow keys to move from one file icon to another and then use the ENTER key to select the file you want.

To copy and move files, you can use the standard drag-and-drop method with your mouse. To copy a file, you locate it by using the file manager. Open another file manager window to the directory to which you want the file copied. Then click and drag the File icon to that window. A pop-up menu appears with selections for Move, Copy, or Link. Choose Copy. To move a file to another directory, follow the same procedure, but select Move from the pop-up menu. To copy or move a directory, use the same procedure as for files. All the directory's files and subdirectories are also copied or moved.

To rename a file, click its icon and press F2, or right-click the icon and select Rename from the pop-up menu. The name below the icon will become boxed, editable text that you can then change.

You delete a file either by removing it immediately or placing it in a Trash folder to delete later. To delete a file, select it and then choose the Delete entry in the Edit menu. You can also right-click the icon and select Delete. To place a file in the Trash folder, click and drag it to the Trash icon on your desktop or select Move To Trash from the Edit menu. You can later open the Trash folder and delete the files. To delete all the files in the Trash folder, right-click the Trash icon and select Empty Trash Bin from the pop-up menu. To restore any files in the Trash bin, open the Trash bin and drag them out of the Trash folder.

Each file or directory has properties associated with it that include permissions, the filename, and its directory. To display the Properties window for a given file, right-click the file's icon and select the Properties entry. On the General panel, you see the name of the file displayed. To change the file's name, replace the name there with a new one. Permissions are set on the Permissions panel. Here, you can set read, write, and execute permissions for user, group, or other access to the file. The Group entry enables you to change the group for a file. The Meta Info panel lists information specific to that kind of file, for example, the number of lines and characters in a text file. An image file will list features such as resolution, bit depth, and color.

TIP *KDE automatically searches for and reads an existing* ***.directory*** *file located in a directory. A* ***.directory*** *file holds KDE configuration information used to determine how the directory is displayed. You can create such a file in a directory and place a setting in it to set display features, such as the icon to use to display the directory folder.*

Web and FTP Access

The KDE file manager also doubles as a full-featured web browser and an FTP client. It includes a box for entering either a pathname for a local file or a URL for a web page on the Internet or your intranet. A navigation toolbar can be used to display previous web pages or previous directories. The Home button will always return you to your home directory. When accessing a web page, the page is displayed as on any web browser. With the navigation toolbar, you can move back and forth through the list of previously displayed pages in that session.

The KDE file manager also operates as an FTP client. When you access an FTP site, you navigate the remote directories as you would your own. The operations to download a file are the same as copying a file on your local system. Just select the file's icon or entry in the file manager window and drag it to a window showing the local directory to which you want it downloaded. Then, select the Copy entry from the pop-up menu that appears. Konqueror also includes KSSL, which provides full SSL support for secure connections, featuring a secure connection status display.

TIP *KDE features the KGet tool for Konqueror, which manages FTP downloads, letting you select, queue, suspend, and schedule downloads, while displaying status information on current downloads.*

Configuring Konqueror

As a file browser, web and FTP browser, and integral part of the KDE desktop, Konqueror has numerous configuration options. To configure Konqueror, open the Configure Konqueror window by selecting Configure Konqueror from a Konqueror window Settings menu (see Figure 9-5). This window displays a category listing on a sidebar. The initial categories deal with basic file management options like appearance, behavior, previews, and file associations. In Behavior, you specify such actions as displaying tooltips and opening folders in new windows. Appearance lets you select the font and font size. With Previews you can set the size of previewed icons, as well as specify the kind of files to retrieve metadata on. File Associations lets you set default applications for different kinds of files (same as File Association in KDE Components in Control Center).

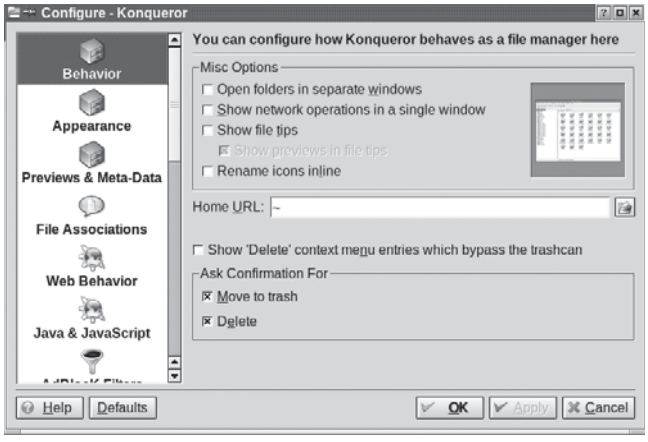


FIGURE 9-5 Configure Konqueror window

The remaining categories deal with web browser configurations, including configuring proxies and web page displays, as well as such basic behavior as highlighting URLs, fonts to use, managing cookies, and selecting encryption methods. The History category lets you specify the number of history items and their expiration date. With the Plugins category you can see a listing of current browser plug-ins as well as scan for new ones.

KDE Configuration: KDE Control Center

With the KDE Control Center, you can configure your desktop and system, changing the way it is displayed and the features it supports (see Figure 9-6). The Control Center can be directly started by selecting Control Center from the main menu.

The Control Center window is divided into two panes. The left pane shows a tree view of all the components you can configure, and the right pane displays the dialog windows for the selected component. See the Help viewer for a current listing of K Desktop configuration modules.

On the left pane, components are arranged into categories whose titles you can expand or shrink. The Internet & Network heading holds entries for configuring the KDE file manager's network tools, including web browser features, Samba (Windows) access, and wireless connectivity. Under Appearances Themes, you can set different features for displaying and controlling your desktop. For example, the Background entry enables you to select a different background color or image for each one of your virtual desktops. Other entries enable you to configure components such as the screen saver, the language used, and the window style. The Peripherals heading holds entries that let you configure your mouse, keyboard, and printer. The Sound & Multimedia heading contains panels for configuring sound components. From the Control Center, you can also access a set of specialized KDE system configuration tools. Currently these include a login manager and a font manager.

The KDE Components category configures the behavior of your KDE interface. The Component Chooser lets you choose default components for applications, including the mail client, the terminal tool, and web browser. File Associations associates file MIME types with

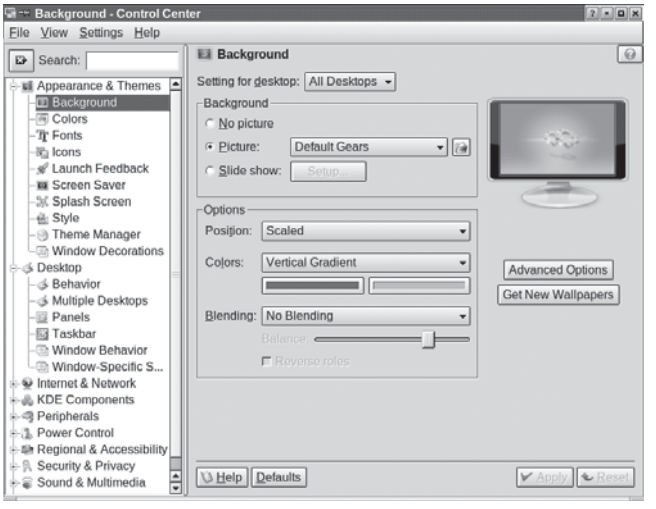


Figure 9-6
KDE Control Center

default applications. The File Manager entry lets you set file manager features such as the font used and the files to preview. With the Session Manager, you can configure session startup and shutdown actions, for instance, restoring previous sessions on startup or automatically shutting down the system when you exit KDE. The Service Manager will list KDE daemons, both those loaded on demand and those on startup. You can elect whether to have a daemon run at startup, as well as manually start and stop daemons. Currently, KDE file sharing and Internet daemons are started automatically, but you can elect instead to have them turned off and start them manually when you want that kind of connectivity.

You can also access the Control Center entries from any file manager window (see Figure 9-7). Select Settings from the file manger Go menu. This opens a folder listing icons for all the KDE configuration categories as icons and folders. KDE configuration uses **the URL settings:/**.

.kde and Desktop User Directories

Your **.kde** directory holds files and directories used to maintain your KDE desktop. As with GNOME, the **Desktop** directory holds KDE desktop files whose icons are displayed on the desktop. Configuration files are located in the **.kde/share/config** directory. Here you can find the general configuration files for different KDE components: **kwinrc** holds configuration commands for the window manager, **kmailrc** for mail, and **kickerrc** for your panel, while **kdeglobals** holds keyboard shortcuts along with other global definitions. You can place configuration directives directly in any of these files; **.kde/share/mimelnk** holds the desktop files for the menu entries added by the user. The **.kde/share/apps** directory contains files and directories for configuring KDE applications, including **koffice**, **kmail**, and even **konqueror**.

MIME Types and Associated Applications

As you install new kinds of programs, they may use files of a certain type. In that case, you will need to register the type with KDE so that it can be associated with a given application or group of applications. For example, the MIME type for GIF images is **image/gif**, which is associated with image-viewing programs. You use the KDE Control Center to set up a new MIME type or to change MIME type associations with applications. Select the File Association entry under KDE Components. This will list known MIME types and their associated filename extensions. Select an entry to edit it, where you can change the applications associated with it. KDE saves its MIME type information in a separate file called **mimelnk** in the KDE configuration directory.

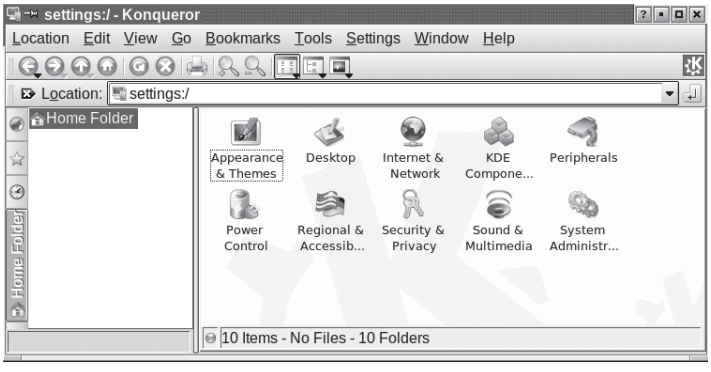


Figure 9-7 System Settings access (Control Center) from the file manager

KDE Directories and Files

When KDE is installed on your system, its systemwide application, configuration, and support files may be installed in the same system directories as other GUIs and user applications. On Red Hat Enterprise Linux and Fedora, KDE is installed in the standard system directories with some variations, such as **/usr/bin** for KDE program files, **/usr/lib/kde3**, which holds KDE libraries, and **/usr/include/kde**, which contains KDE header files used in application development.

The directories located in the **share** directory contain files used to configure system defaults for your KDE environment (the system **share** directory is located at **/usr/share**). The **share/mimelnk** directory maps its files to KDE icons and specifies MIME type definitions. Their contents consist of desktop files having the extension **.desktop**, one for each menu entry. The **share/apps** directory contains files and directories set up by KDE applications; **share/config** contains the configuration files for particular KDE applications. These are the systemwide defaults that can be overridden by users' own configurations in their own **.kde/share/config** directories. The **share/icons** directory holds the default icons used on your KDE desktop and by KDE applications, as well as for the Bluecurve interface. As noted previously, in the user's home directory, the **.kde** directory holds a user's own KDE configuration for the desktop and its applications.

Each user has a **Desktop** directory that holds KDE link files for all icons and folders on the user's desktop (see Table 9-3). These include the Trash folders and the CD-ROM and home directory links.

System KDE Directory	Description
/usr/bin	KDE programs
/usr/lib/kde3	KDE libraries
/usr/include/kde	Header files for use in compiling and developing KDE applications
/usr/share/config	KDE desktop and application configuration files
/usr/share/mimelnk	Desktop files used to build the main menu
/usr/share/apps	Files used by KDE applications
/usr/share/icons	Icons used in KDE desktop and applications
/usr/share/doc	KDE Help system
User KDE Directory	**Description**
.kde/AutoStart	Applications automatically started up with KDE
.kde/share/config	User KDE desktop and application configuration files for user-specified features
.kde/share/mimelnk	Desktop files used to build the user's menu entries on the KDE main menu
.kde/share/apps	Directories and files used by KDE applications
Desktop	Desktop files for icons and folders displayed on the user's KDE desktop
Desktop/Trash	Trash folder for files marked for deletion

TABLE 9-3 KDE Installation Directories

PART IV

Linux Software

CHAPTER 10
Software Management

CHAPTER 11
Office and Database Applications

CHAPTER 12
Graphics Tools and Multimedia

CHAPTER 13
Mail and News Clients

CHAPTER 14
Web, FTP, and Java Clients

CHAPTER 15
Network Tools

This page intentionally left blank

CHAPTER 10

Software Management

Installing, uninstalling, or updating software packages has always been a simple process in Linux because of the widespread use of package formats like the Red Hat Package Manager (RPM) or the Debian package manager (DEB). Instead of using a standard tar archive, software is packaged in an archive using a special format that can be managed using a package manager. A package manager archive contains all the program files, configuration files, data files, and even documentation that constitutes a software application. With one simple operation, the package manager installs all these for you. It also checks for any other software packages that the program may need to run correctly. You can even create your own packages.

Many Linux distributions now manage software packages using online repositories. All software is downloaded directly and installed using the distribution software installer and updater. This approach heralds a move from thinking of most Linux software as included on a few disks, to viewing the disk as just a core from which you can expand your installed software as you like from online repositories. With the integration of software repository access into your Linux system, you can now think of that software as an easily installed extension of your current collection. Relying on disk media for your software becomes, in a sense, obsolete.

You can also download source code versions of applications and then compile and install them on your system. Although this process once was complex, it has been significantly streamlined with the addition of *configure scripts.* Most current source code, including GNU software, is distributed with a configure script. The configure script automatically detects your system configuration and generates a *Makefile,* which is used to compile the application and create a binary file that is compatible with your system. In most cases, with a few Makefile operations you can compile and install complex source code on any system.

You can download Linux software from many online sources. You can find sites for particular kinds of applications, such as GNOME and KDE, as well as for particular distributions. With distribution-enabled repositories, Linux can automatically download and update software installed from software packages.

Software Package Types

The software packages on RPM sites like **freshrpms.net** and **rpmfind.net** will have the file extension **.rpm**. RPM packages that contain source code have an extension **.src.rpm**. Other packages, such as those in the form of source code that you need to compile, come in

Extension	File
.rpm	A software package created with the Red Hat Software Package Manager, used on Fedora, Red Hat, Centos, and SuSE distributions
.deb	A Debian Linux package
.src.rpm	Software packages that are source code versions of applications, created with the Red Hat Software Package Manager
.gz	A **gzip**-compressed file (use **gunzip** to decompress)
.bz2	A **bzip2**-compressed file (use **bunzip2** to decompress; also use the **j** option with **tar**, as in **xvjf**)
.tar	A tar archive file (use **tar** with **xvf** to extract)
.tar.gz	A **gzip**-compressed `tar` archive file (use **gunzip** to decompress and **tar** to extract; use the **z** option with **tar**, as in **xvzf**, to both decompress and extract in one step)
.tar.bz2	A **bzip2**-compressed `tar` archive file (extract with **tar -xvzj)**
.tz	A `tar` archive file compressed with the **compress** command
.Z	A file compressed with the **compress** command (use the **decompress** command to decompress)
.bin	A self-extracting software file
.torrent	A BitTorrent file for performing BitTorrent-distributed downloads (torrent information only)

Table 10-1 Linux Software Package File Extensions

a variety of compressed archives. These commonly have the extension **.tar.gz**, **.tgz**, or **.tar.bz2**. They are explained in detail later in the chapter. Table 10-1 lists several common file extensions that you will find for the great variety of Linux software packages available to you.

Downloading ISO and DVD Distribution Images with BitTorrent

Very large files like distribution ISO images can be downloaded using BitTorrent. BitTorrent is a distributed download operation, where many users on the Internet participate in the same download, each uploading parts that others can in turn download. The file is cut into small IP packets, and each packet is individually uploaded and downloaded as if it were a separate file. Your BitTorrent client will automatically combine the packets into the complete file. There is no shared disk space as in file sharing methods. No access is granted to other uses. A user simply requests that others send him or her a packet. It is strictly a transmission operation, as if many users were participating in the same transmission instead of just one.

You will need to use a BitTorrent client. You have several BitTorrent clients to choose from, such as **azureus**, **rtorrent**, **ctorrent**, **ktorrent**, along with the original BitTorrent. For the original BitTorrent, there are two packages, the **bittorrent** and **bittorrent-gui**, which provides a GNOME interface. The original BitTorrent site is now a commercial site for downloading movies that require Windows Media Player and digital rights management (DRM).

To start a torrent, just click the torrent entry for a file on your web browser. Firefox will prompt you to choose whether to start up the application directly or download the file. If you run the application, the BitTorrent client will be started and your download will begin. You can stop at any time and restart the torrent later. It will automatically start up where you left off, keeping what you have downloaded so far. The BitTorrent client will automatically adjust to the appropriate download/upload scale, but you can adjust this as you wish. There are buttons for pausing and stopping the download, as well as for obtaining detailed information about the torrent. An icon bar shows the progress and the estimated time remaining, though this may shorten as the download progresses. The client will show all torrents you have in process, showing how much is downloaded for each and letting you choose which you want to be active. Configuration entries will let you adjust such behavior as what port to use, the default download directory, and whether to allow torrents to run in parallel.

The torrent file that you download is not the ISO or DVD image. That will be downloaded by BitTorrent. Instead this is a simple, small BitTorrent file that holds information on how to access and start this particular torrent. You can first download the torrent file and then later start it to start up the torrent download. You can have a collection of torrent files that you can start and stop as you want. You will have a small torrent file of type **.torrent** on your disk. Double-click it to start a BitTorrent client and the download torrent for the distribution binary ISO images.

You can use BitTorrent for any file for which you can find an associated torrent file, but downloading is time-sensitive in that it depends on how many users are participating in the torrent. The Linux distribution torrents are usually well maintained, with several users providing constant upload support, known as seeds. Others may die out, if there are no other users participating in the torrent. The speed of the download is directly dependent on the number of users participating in the torrent. This is why the torrent is much more suitable to time-specific distributions, such as when a Linux distribution provides a new release.

NOTE *The BitTorrent package also provides the tools for creating your own torrent to distribute a file. Two distribution methods are now available, tracker and trackerless. A trackerless method requires no server support.*

Red Hat Package Manager (RPM)

Several Linux distributions, including Fedora, Red Hat, and openSUSE, use RPM to organize Linux software into packages you can automatically install, update, or remove. RPM is a command line–driven package management system that is capable of installing, uninstalling, querying, verifying, and updating software packages installed on Linux systems. An RPM software package operates as its own installation program for a software application. A Linux software application often consists of several files that need to be installed in different directories. The program itself is most likely placed in a directory called **/usr/bin**; online manual files like Man pages go in other directories, and library files go in yet another directory. In addition, the installation may require modification of certain configuration files on your system. The RPM software package performs all these tasks for you. Also, if you later decide you don't want a specific application, you can uninstall packages to remove all the files and configuration information from your system. RPM works similarly to the Windows Install Wizard, automatically installing software, including configuration, documentation,

image, sample, and program files, along with any other files an application may use. All are installed in their appropriate directories on your system. RPM maintains a database of installed software, keeping track of all the files installed. This enables you to also use RPM to uninstall software, automatically removing all files that are part of the application.

NOTE *Red Hat, Fedora, and other Linux distributions that use RPM packages can use Yum (Yellowdog Updater Modified) to automatically download, install, and update software from online RPM repositories (**linux.duke.edu/projects/yum**). Check the Yum documentation for your distribution for details. Installation and updating is a very simple point-and-click operation, with Yum detecting the exact package version you need for your system. Yum can be used for any Yum-compliant repository.*

To install and uninstall RPM packages, you can use the **rpm** command directly from a shell prompt. Although you should download RPM packages for your particular distribution, numerous RPM software packages are designed to run on any Linux system. You can learn more about RPM at its website at **rpm.org** and at **wiki.rpm.org**. The sites contain up-to-date versions and documentation for RPM.

NOTE *RPM has been reorganized as an independent project and is no longer considered just a Red Hat tool.*

The naming conventions for RPM packages vary from one distribution to another. The package name includes the package version, along with its platform (**i386** for Intel PCs) and the **.rpm** extension. An example of the Emacs editor's RPM package for Intel systems is shown here:

```
emacs-21.4-3.i386.rpm
```

TIP *RPM packages with the term **noarch** are used for architecture-independent packages. This means that they are designed to install on any Linux system. Packages without **noarch** may be distribution- or architecture-dependent.*

The rpm Command

With the **rpm** command, you can maintain packages, query them, build your own, and verify the ones you have. Maintaining packages involves installing new ones, upgrading to new versions, and uninstalling packages. The **rpm** command uses a set of options to determine what action to take. In addition, certain tasks, such as installing or querying packages, have their own options that further qualify the kind of action they take. For example, the **-q** option queries a package, but when combined with the **-l** option, it lists all the files in that package. Table 10-2 lists the set of **rpm** options. The syntax for the **rpm** command is as follows (*rpm-package-name* is the name of the software package you want to install):

```
rpm options rpm-package-name
```

A complete description of **rpm** and its capabilities is provided in the online manual:

```
# man rpm
```

Mode of Operation	Effect
rpm -i*options package-file*	Installs a package; the complete name of the package file is required.
rpm -e*options package-name*	Uninstalls (erases) a package; you only need the name of the package, often one word.
rpm -q*options package-name*	Queries a package. An option can be a package name, a further option and package name, or an option applied to all packages.
rpm -U*options package-name*	Upgrades; same as install, but any previous version is removed.
rpm -F*options package-name*	Upgrades, but only if package is currently installed.
rpm -verify*options*	Verifies a package is correctly installed; uses same options as query. You can use **-V** or **-y** in place of **-verify**.
--percent	Displays percentage of package during installation.
--replacepks	Installs an already-installed package.
--replacefiles	Replaces files installed by other packages.
--redhatprovides *dependent-files*	Searches for dependent packages.
--oldfiles	Installs an older version of a package already installed.
--test	Tests installation; does not install, only checks for conflicts.
-h	Displays # symbols as package is installed.
--excludedocs	Excludes documentation files.
--nodeps	Installs without doing any dependency checks (dangerous).
--force	Forces installation despite conflicts (dangerous).
Uninstall Options (to be Used with -e)	
--test	Tests uninstall. Does not remove, only checks for what is to be removed.
--nodeps	Uninstalls without checking for dependencies.
--allmatches	Removes all versions of package.
Query Options (to be used with -q)	
package-name	Queries package.
-qa	Queries all packages.
-qf *filename*	Queries package that owns *filename*.

Table 10-2 Red Hat Package Manager (RPM) Options

Query Options (to be used with -q)	
-qR	List packages on which this package depends.
-qp *package-name*	Queries an uninstalled package.
-qi	Displays all package information.
-ql	Lists files in package.
-qd	Lists only documentation files in package.
-qc	Lists only configuration files in package.
-q --dump	Lists only files with complete details.
General Options (to be used with any option)	
-vv	Debugs; displays descriptions of all actions taken.
--quit	Displays only error messages.
--version	Displays RPM version number.
--help	Displays detailed use message.
--root*directory*	Uses directory as top-level directory for all operations (instead of root).
--dbpath*directory*	Uses RPM database in the specified directory.
--dbpath *cmd*	Pipes output of RPM to the command **cmd**.
--rebuilddb	Rebuilds the RPM database; can be used with the **-root** and **-dbpath** options.
--initdb	Builds a new RPM database; can be used with the **-root** and **-dbpath** options.
Other Sources of Information	
rpm.org	The RPM website with detailed documentation.
RPM Man page (**man rpm**)	Detailed list of options.

TABLE 10-2 Red Hat Package Manager (RPM) Options (*continued*)

Querying Information from RPM Packages and Installed Software

The **-q** option tells you if a package is already installed, and the **-qa** option displays a list of all installed packages. Piping this output to a pager utility, such as **more**, is best.

```
# rpm -qa | more
```

In the next example, the user checks to see if Emacs is already installed on the system. Notice the full filename of the RPM archive is unnecessary. If the package is installed, your system has already registered its name and where it is located.

```
# rpm -q emacs
emacs-22.0.95-1
```

Option	Meaning
`-q` *application*	Checks to see if an application is installed.
`-qa` *application*	Lists all installed RPM applications.
`-qf` *filename*	Lists applications that own *filename.*
`-qR` *application*	Lists applications on which this application depends.
`-qi` *application*	Displays all application information.
`-ql` *application*	Lists files in the application.
`-qd` *application*	Lists only documentation files in the application.
`-qc` *application*	Lists only configuration files in the application.

TABLE 10-3 Query Options for Installed Software

You can combine the `q` option with the `i` or `l` option to display information about the package. The option `-qi` displays information about the software, such as the version number or author (`-qpi` queries an uninstalled package file). The option `-ql` displays a listing of all the files in the software package. The `--h` option provides a complete list of `rpm` options. Common query options are shown in Table 10-3.

To display information taken directly from an RPM package, you add the `p` qualifier to the `q` options as shown in Table 10-4. The `-qpi` combination displays information about a specific package, and `-qpl` displays a listing of the files a given RPM package contains. In this case, you must specify the entire filename of the RPM package. You can avoid having to enter the entire name simply by entering a unique part of the name and using the `*` filename-matching character to generate the rest.

If your RPM query outputs a long list of data, like an extensive list of files, you can pipe the output to the `less` command to look at it screen by screen, or even redirect the output to a file.

```
# rpm -ql emacs | less
# rpm -qpl emacs-22.0.95-1.i386.rpm  > mytemp
```


Option	Meaning
`-qpi` *RPM-file*	Displays all package information in the RPM package.
`-qpl` *RPM-file*	Lists files in the RPM package.
`-qpd` *RPM-file*	Lists only documentation files in the RPM package.
`-qpc` *RPM-file*	Lists only configuration files in the RPM package.
`-qpR` *RPM-file*	Lists packages on which this RPM package depends.

TABLE 10-4 Query Options for RPM Packages

Installing and Updating Packages with rpm

You use the **-i** option to install new packages and the **-U** option to update currently installed packages with new versions. With an **-e** option, **rpm** uninstalls the package. If you try to use the **-i** option to install a newer version of an installed package, you will receive an error saying the package is already installed. When a package is installed, RPM checks its signature, using imported public keys from the software vendor. If the signature check fails, an error message is displayed, specifying NOKEY if you do not have the appropriate public key. If you want to install over an already-installed package, you can force installation with the **--replacepks** option. Sometimes a package will include a file, such as a library, that is also installed by another package. To allow a package to overwrite the file installed by another package, you use the **--replacefiles** option. Many packages depend on the libraries installed by other packages. If these dependent packages are not already installed, you will first have to install them. RPM informs you of the missing dependent files and suggests packages to install. If no packages are suggested, you can use the **--redhatprovides** option with the missing files to search for needed packages.

The **-U** option also installs a package if it is not already installed, whereas the **-F** option will only update installed packages. If the package includes configuration files that will overwrite currently installed configuration files, it will save a copy of each current configuration file in a file ending with **.rpmsave**, such as **/etc/mtools.conf.rpmsave**. This preserves any customized configuration changes you may have made to the file. Be sure to also check for configuration compatibilities between the previous and updated versions. If you are trying to install a package that is older than the one already installed, then you need to use the **--oldpackages** option.

```
# rpm -Uvh emacs-22.0.95-1.i386.rpm
```

Removing RPM Software Packages

To remove a software package from your system, first use **rpm -q** to make sure it is actually installed. Then use the **-e** option to uninstall it. You needn't use the full name of the installed file; you only need the name of the application. For example, if you decide you do not need Gnumeric, you can remove it using the **-e** option and the software name, as shown here:

```
# rpm  -e  gnumeric
```

RPM: Verifying an RPM Installation

You can use the verify option (**-V**) to see if any problems occurred with the installation. RPM compares the current attributes of installed files with information about them placed in the RPM database when the package was installed. If no discrepancies exist, RPM outputs nothing. Otherwise, RPM outputs a sequence of eight characters, one for each attribute, for each file in the package that fails. Those that do not differ have a period. Those that do differ have a corresponding character code, as shown in Table 10-5.

The following example verifies the ProFTPD package:

```
[root@turtle mypackages]# rpm -V proftpd
```

Attribute	Explanation
5	MD5 checksum
S	File size
L	Symbolic link
T	File modification time
D	Device
U	User
G	Group
M	Mode (includes permissions and file types)

TABLE 10-5 RPM Discrepancy Codes

To compare the installed files directly with the files in an RPM package file, you use the **-Vp** option, much like the **-qp** option. To check all packages, use the **-Va** option as shown here:

```
# rpm -Va
```

If you want to verify a package, but you only know the name of a file in it, you can combine verify with the **-f** option. The following example verifies the RPM package containing the **ftp** command:

```
# rpm -Vf  /bin/ftp
```

Rebuilding the RPM Database

RPM maintains a record of the packages it has installed in its **RPM** database. You may, at times, have to rebuild this database to ensure RPM has current information on what is installed and what is not. Use the **--rebuilddb** option to rebuild your database file:

```
#  rpm --rebuilddb
```

To create a new RPM database, use the **--initdb** option. This option can be combined with **--dbpath** to specify a location for the new database.

Debian

Among most Linux distributions, there are basically two major software packaging, RPM and DEB, used primarily in the Debian and Ubuntu distributions. The DEB format is much more capable than its RPM counterpart. For, example, a Debian package will automatically resolve dependencies, installing any other needed packages instead of simply reporting their absence, like RPM does. Debian also uses a different package naming format than RPM. Packages are named with the software name, the version number, and the **.deb** extension. Check **debian.org/doc** for more information.

Two basic package managers are available for use with Debian packages: the Advanced Package Tool (APT) and the Debian Package tool (dpkg). For APT, you use the **apt-get** tool to manage your packages. **apt-get** can even download packages as well as compile source code versions for you. The **apt-get** tool takes two arguments: the command to perform and the name of the package. Other APT package tools follow the same format. The command is a term such as **install** for installing packages or **remove** to uninstall a package. To install the kernel image package you would use:

```
apt-get install kernel-image-2.6.21.deb
```

Upgrading is a simple matter of using the **upgrade** command. With no package specified, apt-get with the **upgrade** command will upgrade your entire system, downloading from an FTP site or copying from a CD-ROM and installing packages as needed. Add the **-u** option to list packages as they are upgraded.

```
apt-get -u upgrade
```

You can even upgrade to a new release with the **dist-upgrade** command.

```
apt-get -u dist-upgrade
```

There are several popular front ends for apt-get that let you manage your software easily, like synaptic, gnome-apt, aptitude, and deselect. Configuration for APT is held in the **/etc/apt** directory. Here the **sources.list** file lists the distribution repositories from where packages are installed. Source lists for additional third-party repositories (like that for Wine) are kept in the /**etc/sources.list.d** directory. GPG database files hold validation keys for those repositories. Specific options for apt-get are kept in either an **/etc/apt.conf** file or in various files located in the **/etc/apt.conf.d** directory.

You can also use the **dpkg** tool to manage software, though it is used primarily to obtain information about a package. Its more complex version, **dpkg-deb**, is used to construct Debian packages. The dpkg configuration files are located in the **/etc/dpkg** directory. Configuration is held in the **dpkg.cfg** file and sources in the **origins** directory.

Installing Software from Compressed Archives: .tar.gz

Linux software applications in the form of source code are available at different sites on the Internet. You can download any of this software and install it on your system. Recent releases are often available in the form of compressed archive files. Applications will always be downloadable as compressed archives if they don't have an RPM version. This is particularly true for the recent versions of GNOME or KDE packages. RPM packages are only intermittently generated.

Decompressing and Extracting Software in One Step

Though you can decompress and extract software in separate operations, you will find that the more common approach is to perform both actions with a single command. The **tar** utility provides decompression options you can use to have **tar** first decompress a file for you, invoking the specified decompression utility. The **z** option automatically invokes

gunzip to unpack a **.gz** file, and the **j** option unpacks a **.bz2** file. Use the **Z** option for **.Z** files. For example, to combine the decompressing and unpacking operation for a **.tar.gz** file into one **tar** command, insert a **z** option into the option list, **xzvf** (see the later section "Extracting Software" for a discussion of these options). The next example shows how you can combine decompression and extraction in one step:

```
# tar xvzf htdig-3.1.6.tar.gz
```

For a -compressed archive, you use the **j** option instead of the **z** option.

```
# tar xvjf htdig-3.1.6.tar.bz2
```

Decompressing Software Separately

Many software packages under development or designed for cross-platform implementation may not be in an RPM format. Instead, they may be archived and compressed. The filenames for these files end with the extension **.tar.gz**, **.tar.bz2**, or **.tar.Z**. The different extensions indicate different decompression methods using different commands: **gunzip** for **.gz**, **bunzip2** for **.bz2**, and **decompress** for **.Z**. In fact, most software with an RPM format also has a corresponding **.tar.gz** format. After you download such a package, you must first decompress it and then unpack it with the **tar** command. The compressed archives can hold either source code that you then need to compile or, as is the case with Java packages, binaries that are ready to run.

A *compressed archive* is an archive file created with **tar** and then compressed with a compression tool like **gzip**. To install such a file, you must first decompress it with a decompression utility like **gunzip** and then use **tar** to extract the files and directories making up the software package. Instead of the **gunzip** utility, you could also use **gzip -d**. The next example decompresses the **htdig-3.2.6.tar.gz** file, replacing it with a decompressed version called **htdig-3.2.6.tar**:

```
# ls
 htdig-3.2.6.tar.gz
# gunzip htdig-3.2.6.tar.gz
# ls
htdig-3.2.6.tar
```

You can download compressed archives from many different sites, including those mentioned previously. Downloads can be accomplished with FTP clients such as NcFTP and gFTP, or with any web browser. Once downloaded, any file that ends with **.Z**, **.bz2**, **.zip**, or .**gz** is a compressed file that must be decompressed.

For files ending with **.bz2**, you use the **bunzip2** command. The following example decompresses a **bz2** version:

```
# ls
 htdig-3.2.6.tar.bz2
# bunzip2 htdig-3.2.6.tar.bz2
# ls
htdig-3.2.6.tar
```

Files ending with **.bin** are self-extracting archives. Run the bin file as if it were a command. You may have to use **chmod** to make it executable. The j2sdk software package is currently distributed as a self-extracting bin file.

```
# j2sdk-1.4.2-FCS-linux-i386.tar.bin
# ls
j2sdk-1.3.0-FCS-linux-i386.tar
```

Selecting an Install Directory

Before you unpack the archive, move it to the directory where you want it. Source code packages are often placed in a directory like **/usr/local/src**, and binary packages go in designated directories. When source code files are unpacked, they generate their own subdirectories from which you can compile and install the software. Once the package is installed, you can delete this directory, keeping the original source code package file (**.tar.gz**).

Packages that hold binary programs ready to run, like Java, are meant to be extracted in certain directories. Usually this is the **/usr/local** directory. Most archives, when they unpack, create a subdirectory named with the application name and its release, placing all those files or directories making up the software package into that subdirectory. For example, the file **htdig-3.2.6.tar** unpacks to a subdirectory called **htdig-3.2.6**. In certain cases, the software package that contains precompiled binaries is designed to unpack directly into the system subdirectory where it will be used. For example, it is recommended that **j2sdk-1.4.2-FCS-linux-i386.tar** be unpacked in the **/usr/local** directory, where it will create a subdirectory called **j2sdk-1.4.2**. The **/usr/local/j2sdk-1.4.2/bin** directory holds the Java binary programs.

Extracting Software

First, use **tar** with the **t** option to check the contents of the archive. If the first entry is a directory, then when you extract the archive, that directory is created and the extracted files are placed in it. If the first entry is not a directory, you should first create one and then copy the archive file into it. Then extract the archive within that directory. If no directory exists as the first entry, files are extracted to the current directory. You must create a directory yourself to hold these files.

```
# tar tvf htdig-3.1.6.tar
```

Now you are ready to extract the files from the tar archive. You use **tar** with the **x** option to extract files, the **v** option to display the pathnames of files as they are extracted, and the **f** option, followed by the name of the archive file:

```
# tar xvf htdig-3.2.6.tar
```

You can also decompress and extract in one step using the **-z** option for **gz** files and **-j** for **bz2** files.

```
# tar xvzf htdig-3.1.6.tar.gz
```

The extraction process creates a subdirectory consisting of the name and release of the software. In the preceding example, the extraction created a subdirectory called **htdig-3.2.6**.

You can change to this subdirectory and examine its files, such as the **readme** and **INSTALL** files.

```
# cd htdig-3.2.6
```

Installation of your software may differ for each package. Instructions are usually provided along with an installation program. Be sure to consult the **readme** and **INSTALL** files, if included. See the following section on compiling software for information on how to create and install the application on your system.

Compiling Software

Some software may be in the form of source code that you need to compile before you can install it. This is particularly true of programs designed for cross-platform implementations. Programs designed to run on various Unix systems, such as Sun, as well as on Linux, may be distributed as source code that is downloaded and compiled in those different systems. Compiling such software has been greatly simplified in recent years by the use of configuration scripts that automatically detect a given system's hardware and software configuration, and then allow you to compile the program accordingly. For example, the name of the C compiler on a system could be **gcc** or **cc**. Configuration scripts detect which is present and select it for use in the program compilation.

A configure script works by generating a customized Makefile, designed for that particular system. A Makefile contains detailed commands to compile a program, including any preprocessing, links to required libraries, and the compilation of program components in their proper order. Many Makefiles for complex applications may have to access several software subdirectories, each with separate components to compile. The use of configure and Makefile scripts vastly automates the compile process, reducing the procedure to a few simple steps.

First, change to the directory where the software's source code has been extracted:

```
# cd /usr/local/src/htdig-3.2.6
```

Before you compile software, read the **readme** or **INSTALL** files included with it. These give you detailed instructions on how to compile and install this particular program.

Most software can be compiled and installed in three simple steps. Their fist step is the **./configure** command, which generates your customized Makefile. The second step is the **make** command, which uses a Makefile in your working directory (in this case, the Makefile you just generated with the **./configure** command) to compile your software. The final step also uses the **make** command, but this time with the **install** option. The Makefile generated by the **./configure** command also contains instructions for installing the software on your system. Using the **install** option runs just those installation commands. To perform the installation, you have to be logged in as the root user, giving you the ability to add software files to system directories as needed. If the software uses configuration scripts, compiling and installing usually involves only the following three simple commands:

```
# ./configure
#  make
#  make install
```

In the preceding example, the **./configure** command performs configuration detection. The **make** command performs the actual compiling, using a Makefile script generated by the **./configure** operation. The **make install** command installs the program on your system, placing the executable program in a directory, such as **/usr/local/bin**, and any configuration files in **/etc**. Any shared libraries it created may go into **/usr/local/lib**.

Once you have compiled and installed your application, and you have checked that it is working properly, you can remove the source code directory that was created when you extracted the software. You can keep the tar archive file in case you need to extract the software again. Use **rm** with the **-rf** options so that all subdirectories will be deleted and you do not have to confirm each deletion.

TIP *Be sure to remember to place the period and slash before the* ***configure*** *command. The* **./** *references a command in the current working directory, rather than another Linux command with the same name.*

Configure Command Options

Certain software may have specific options set up for the **./configure** operation. To find out what these are, you use the **./configure** command with the **--help** option:

```
# ./configure --help
```

A useful common option is the **-prefix** option, which lets you specify the install directory:

```
# ./configure -prefix=/usr/bin
```

TIP *Some older X applications use* ***xmkmf*** *directly instead of a configure script to generate the needed Makefile. Although* ***xmkmf*** *has been officially replaced, in this case, enter the command* ***xmkmf*** *in place of* ***./configure****. Be sure to consult the* ***INSTALL*** *and* ***readme*** *files for the software.*

Development Libraries

If you are compiling an X GNOME- or KDE-based program, be sure their development libraries have been installed. For X applications, be sure the **xmkmf** program is also installed. If you chose a standard install when you installed your distribution system, these most likely were not installed. For distributions using RPM packages, these come in the form of a set of development RPM packages, usually with the word "development" or "develop" in their names. You need to install them using **rpm**. GNOME, in particular, has an extensive set of RPM packages for development libraries. Many X applications need special shared libraries. For example, some applications may need the **xforms** library or the **qt** library. Some of these you may need to obtain from online sites.

Shared and Static Libraries

Libraries can be either static, shared, or dynamic. A *static* library is one whose code is incorporated into the program when it is compiled. A *shared* library, however, has its code

loaded for access whenever the program is run. When compiled, such a program simply notes the libraries it needs. Then when the program is run, that library is loaded and the program can access its functions. A *dynamic* library is a variation on a shared library. Like a shared library, it can be loaded when the program is run. However, it does not actually load until instructions in the program tell it to. It can also be unloaded as the program runs, and another library can be loaded in its place. Shared and dynamic libraries make for much smaller code. Instead of a program including the library as part of its executable file, it only needs a reference to it.

Libraries made available on your system reside in the **/usr/lib** and **/lib** directories. The names of these libraries always begin with the prefix **lib** followed by the library name and a suffix. The suffix differs, depending on whether it is a static or shared library. A shared library has the extension **.so** followed by major and minor version numbers. A static library simply has the **.a** extension. A further distinction is made for shared libraries in the old **a.out** format. These have the extension **.sa**. The syntax for the library name is the following:

```
libname.so.major.minor
libname.a
```

The *libname* can be any string, and it uniquely identifies a library. It can be a word, a few characters, or even a single letter. The name of the shared math library is **libm.so.5**, where the math library is uniquely identified by the letter **m** and the major version is **5**, and **libm.a** is the static math library. The name of the X Window library is **libX11.so.6**, where the X Window library is uniquely identified with the letters **X11** and its major version is 6.

Most shared libraries are found in the **/usr/lib** and **/lib** directories. These directories are always searched first. Some shared libraries are located in special directories of their own. A listing of these is placed in the **/etc/ld.conf** configuration file. These directories will also be searched for a given library. By default, Linux first looks for shared libraries, then static ones. Whenever a shared library is updated or a new one installed, you need to run the **ldconfig** command to update its entries in the **/etc/ld.conf** file as well as links to it (if you install from an RPM package, this is usually done for you).

Makefile File

If no configure script exists and the program does not use **xmkmf**, you may have to edit the software's Makefile directly. Be sure to check the documentation for such software to see if any changes must be made to the Makefile. Only a few changes may be necessary, but more detailed changes require an understanding of C programming and how **make** works with it. If you successfully configure the Makefile, you may only have to enter the **make** and **make install** commands. One possible problem is locating the development libraries for C and the X Window System. X libraries are in the **/usr/X11R6/lib** directory. Standard C libraries are located in the **/usr/lib** directory.

Command and Program Directories: PATH

Programs and commands are usually installed in several standard system directories, such as **/bin**, **/usr/bin**, **/usr/X11R6/bin**, or **/usr/local/bin**. Some packages place their commands in subdirectories, however, which they create within one of these standard directories or in an entirely separate directory. In such cases, you may be unable to run those commands

because your system may be unable to locate them in the new subdirectory. Your system maintains a set of directories that search for commands each time you execute one. This set of directories is kept in a system variable called **PATH** that is created when you start your system. If a command is in a directory that is not in this list, your system will be unable to locate and run it. To use such commands, you first need to add the new directory to the set of directories in the **PATH** variable. Installation tools like RPM will automatically update the **PATH** with the appropriate directories for you.

The PATH variable added to by different services that start up when the system boots. A safer approach is to add a **PATH** definition in the **/etc/profile** file using that file's **pathmunge** function, if available.

/etc/profile

To make an application available to all users, you can add the software's directory to the path entry in the **/etc/profile** script. The **/etc/profile** script is a system script executed for each user when the user logs in. Carefully edit the **/etc/profile** file using a text editor, such as KEdit, Gedit, Emacs, or Vi (you may want to make a backup copy first with the **cp** command). On some distributions, you can easily add a directory to the **PATH** variable using the **pathmunge** function, which is also defined in **/etc/profile**. For example, if you install the Java 2 SDK, the Java commands are installed in a subdirectory called **j2sdk-1.4.2/bin** in the **/usr/local** directory. The full pathname for this directory is **/usr/local/j2sdk-1.4.2/bin**. You need to use **pathmunge** to add this directory to the list of directories assigned to **PATH** in the **/etc/profile** file.

```
pathmunge /usr/local/j2sdk-1.4.2/bin
```

You can see other uses of **pathmunge** in **/etc/profile**, such as adding **/sbin** for the root user.

After making your changes, you can execute the **profile** file to have the changes take effect:

```
$  . /etc/profile
```

NOTE *On older systems you had to create a new assignment entry for the* ***PATH*** *variable by adding a line that began with* ***PATH****, followed by an* **=** *sign, the term* ***$PATH****, a colon, and then the directory to be added. The* **$** *before* ***PATH*** *extracted the pathname from the* ***PATH*** *variable. If you added more than one directory, a colon had to separate them and another colon had to be at the end. The following example shows the* ***PATH*** *variable with its list of directories and the* ***/usr/local/j2sdk-1.4.2/bin*** *directory added. Notice the* **$** *before* ***PATH*** *after the* **=** *sign,* ***PATH=$PATH. PATH=$PATH:/usr/local/j2sdk-1.4.2/bin:***

.bash_profile

Individual users can customize their **PATH** variables by placing a **PATH** assignment in either their **.bashrc** or **.bash_profile** file. In this way, users can access commands and programs they create or install for their own use in their own user directories. User **.bash_profile** files already contain the following **PATH** definition. Notice the use of **$PATH**, which keeps all the directories already added to the **PATH** in previous startup scripts like **/etc/profile.**

```
PATH=$PATH:$HOME/bin
```

The following entry in the **.bash_profile** file adds a user's **newbin** directory to the **PATH** variable. Notice both the colon placed before the new directory and the use of the **$HOME** variable to specify the pathname for the user's home directory.

```
PATH=$PATH:$HOME/bin/:$HOME/newbin
```

For the **root** user, the **PATH** definition also includes **sbin** directories. The **sbin** directories hold system administration programs that the root user needs to have access to. The **root** user **PATH** is shown here:

```
PATH=/usr/local/sbin:/usr/sbin:/sbin:$PATH:$HOME/bin
```

Subversion and CVS

The Subversion and the Concurrent Versions System (CVS) are software development methods that allow developers from remote locations to work on software stored on a central server. Subversion is an enhanced version of CVS, designed to eventually replace it. Like CVS, Subversion works with CVS repositories, letting you access software in much the same way. Subversion adds features such as better directory and file access, as well as support for metadata information.

CVS sites allow several developers to work on a file at the same time. This means that they support parallel development, so programmers around the world can work on the same task at the same time through a simple Internet connection. It has become popular among Linux developers as a means of creating software using the Internet. CVS sites are also the source for the most up-to-date versions for different software. Ongoing projects like KDE and GNOME use Subversion or CVS servers to post the most recent versions of their desktop applications, primarily because it is easy to use for program development over the Internet. The **sourceforge.net** site provides a CVS repository for many ongoing Linux Projects. Many CVS sites now support ViewCVS (an enhanced version of webCVS), a web browser front end to a CVS repository that lets you browse and select software versions easily. You can find out more about CVS from **cvshome.org** and about Subversion from **subversion.tigris.org**.

Using a CVS repository for software development involves procedures for accessing a software version, making your changes locally on your system, and then uploading your changed version back to the CVS repository. In effect, you check out software, make your changes in such a way that they are carefully recorded, and then check your version back in to the repository. CVS was originally developed as a front end to the older Revision Control System (RCS) and shares many of the same commands.

Packaging Your Software with RPM

Many research and corporate environments develop their own customized software for distribution within their organization. Sometimes software packages are downloaded and then customized for use in a particular organization. To more easily install such customized software, administrators pack the programs into their own RPM packages. In such packages, you can include your own versions of configuration files, documentation, and modified

source and binaries. RPM automatically installs software on a system in the designated directories, along with any documentation, libraries, or support programs.

The package creation process is designed to take the program through several stages, starting with unpacking it from an archive, then compiling its source code, and finally, generating the RPM package. You can skip any of these stages, up to the last one. If your software is already unpacked, you can start with compiling it. If your software is compiled, you can start with installation. If it is already installed, you can go directly to creating the RPM package.

The build processes for RPM used to be included with the **rpm** command. They are now incorporated into a separate tool called **rpmb**. This tool along with supporting libraries and documentation is located in the rpm-build package. Be sure this package is installed before you try to build RPM packages. You can still run the **rpm** command with the build options, but these are simply aliases for corresponding **rpmb** commands.

11 CHAPTER

Office and Database Applications

A variety of office suites are now available for Linux (see Table 11-1). These include professional-level word processors, presentation managers, drawing tools, and spreadsheets. The freely available versions are described in this chapter. Sun has initiated development of an open source office suite using StarOffice code. The applications, known as OpenOffice.org, provide Office applications integrated with GNOME. OpenOffice .org is currently the primary office application supported by most Linux distributions. KOffice is an entirely free office suite designed for use with KDE. The GNOME Office suite integrates GNOME applications into a productivity suite that is freely available. CodeWeavers CrossOver Office provides reliable support for running MS Office Windows applications directly on Linux, integrating them with KDE and GNOME. You can also purchase commercial office suites such as StarOffice from Sun. For desktop publishing, especially the PDF generation, you can use Scribus, a cross-platform tool available from the Fedora repository.

A variety of database management systems are available for Linux. These include high-powered, commercial-level database management systems, such as Oracle, IBM's DB2, and Sybase. Open source Linux databases are also available, such as MySQL and PostgreSQL. These are among the most widely used on Linux systems. Most of the database management systems available for Linux are designed to support large relational databases. For small personal databases, you can use the desktop database management systems being developed for KDE and GNOME. In addition, some software is available for databases accessed with the Xbase database programming language. These are smaller databases using formats originally developed for dBase on the PC. Various database management systems available to run under Linux are listed in Table 11-6 later in this chapter.

Linux also provides several text editors that range from simple text editors for simple notes to editors with more complex features such as spell-checkers, buffers, or pattern matching. All generate character text files and can be used to edit any Linux text files. Text editors are often used in system administration tasks to change or add entries in Linux configuration files found in the **/etc** directory or a user's initialization or application dot files located in a user's home directory. You can use any text editor to work on source code files for any of the programming languages or shell program scripts.

Website	Description
openoffice.org	OpenOffice.org open source office suite based on StarOffice
koffice.org	KOffice Suite, for KDE
gnome.org/gnome-office	GNOME Office, for GNOME
sun.com/staroffice	StarOffice Suite
codeweavers.com	CrossOver Office (MS Office support)
scribus.net	Scribus desktop publishing tool.

Table 11-1 Linux Office Suites

Running Microsoft Office on Linux: CrossOver

One of the primary concerns for new Linux users is what kind of access they will have to their Microsoft Office files, particularly Word files. The Linux operating system and many applications for it are designed to provide seamless access to MS Office files. The major Linux office suites, including KOffice, OpenOffice.org, and StarOffice, all read and manage any Microsoft Office files. In addition, these office suites are fast approaching the same level of features and support for office tasks as found in Microsoft Office.

If you want to use any Windows application on Linux, three important alternatives are the Wine virtual windows API support, VMware virtual platform technology, and the CrossOver Office by CodeWeavers. VMware and CrossOver are commercial packages.

Wine allows you to run many Windows applications directly, using a supporting virtual windows API. See the Wine website, winehq.com, for a list of supported applications. Well-written applications may run directly from Wine, like the newsbin newsreader. Often you will have to have a working Windows system from which to copy system DLLs needed by particular applications. You can also import your Windows fonts by directly copying them to the Wine font directory. Each user can install their own version of Wine with its own simulated C: partition on which Windows applications are installed. The simulated drive is installed as **drive_c** in your **.wine** directory. The **.wine** directory is a hidden directory. It is not normally displayed with the **ls** command or the GNOME file manager.

In a terminal window, using the **wine** command with an install program will automatically install that Windows application on the simulated C: drive. The following example installs the **newsbin.exe** application.

```
$ wine newsbin.exe
```

Once installed, along with DLLs if needed, an icon will appear on the Linux desktop for the application. The application will start up normally. You can even use any of your Linux directories for your Windows application data files instead of your simulated C: drive.

CrossOver Office is a commercial product that lets you install and run most Microsoft Office applications. CrossOver Office was developed by CodeWeavers, which also supports Windows web browser plug-ins as well as several popular Windows applications like Adobe Photoshop. CrossOver features both standard and professional versions, providing reliable application support. You can find out more about CrossOver Office at **codeweavers.com**.

CrossOver can be installed either for private multiuser mode or managed multiuser mode. In private multiuser mode, each user installs his or her own Windows software, such as full versions of Office. In managed multiuser mode, the Windows software is installed once and all users share it. When you install new software, you first open the CrossOver startup tool, and then on the Add/Remove panel you will see a list of supported software. This will include Office applications as well as some Adobe applications, including earlier versions of Photoshop. An Install Software panel will then let you select whether to install from a CD-ROM or an **.exe** file. For Office on a CD-ROM, select CD-ROM, place the Windows CD-ROM in your CD-ROM drive, and then click Next. The Windows Office installer will start up in a Linux window and will proceed as if you were on a Windows system. When the install requires a restart of the system, CrossOver will simulate it for you. Once the software is installed, you will see a Windows Applications menu on the main menu, from which you can start your installed Windows software. The applications will run within a Linux window, just as if they were running in Windows. You can also try CrossOver for unsupported applications. They may or may not run.

With VMware, you can run Windows under Linux, allowing you to run Windows applications, including Microsoft Office, on your Linux system. For more information, check the VMware website at **vmware.com**.

NOTE *Though Linux allows users to directly mount and access any of the old DOS or FAT32 partitions used for Windows 95, 98, and Me, it can mount NTFS partitions (Windows Vista, XP, 2000, and NT) as read only, with possible write support. There are two drivers for mounting NTFS,* ***ntfs-3g*** *and the original NTFS project support. The* ***ntfs-3g*** *drivers supports writing NTFS partitions.*

OpenOffice.org

OpenOffice.org (OO) is a fully integrated suite of office applications developed as an open source project and freely distributed to all. It is included as the primary office suite for most Linux distributions, accessible from an Office menu. It includes word processing, spreadsheet, presentation, and drawing applications (see Table 11-2). Versions of OpenOffice.org exist for Linux, Windows, and Mac OS. You can obtain information such as online manuals and FAQs, as well as current versions, from the OpenOffice.org website at **openoffice.org**.

Application	Description
Calc	OpenOffice.org spreadsheet
Draw	OpenOffice.org drawing application
Writer	OpenOffice.org word processor
Math	OpenOffice.org mathematical formula composer
Impress	OpenOffice.org presentation manager
Base	Database front end for accessing and managing a variety of different databases

TABLE 11-2 OpenOffice.org Applications

NOTE *Development for OpenOffice.org is being carried out as an open source project called openoffice.org. The core code is based on the original StarOffice. The code developed in the openoffice.org project will then be incorporated into future releases of StarOffice.*

OpenOffice.org is an integrated suite of applications. You can open the writer, spreadsheet, or presentation application directly. Also, in most OpenOffice.org applications, you can select New from the File menu and select a different application if you wish. The Writer word processor supports standard word processing features, such as cut and paste, spell-checker, and text formatting, as well as paragraph styles (see Figure 11-1). You can embed objects within documents, such as using Draw to create figures that you can then drag and drop into the Writer document. You can find out more about each component at their respective product pages listed at **openoffice.org/product**.

Calc is a professional-level spreadsheet. With Math, you can create formulas that you can then embed in a text document. With the presentation manager (Impress), you can create images for presentations, such as circles, rectangles, and connecting elements like arrows, as well as vector-based illustrations. Impress supports advanced features such as morphing objects, grouping objects, and defining gradients. Draw is a sophisticated drawing tool that includes 3-D modeling tools. You can create simple or complex images, including animation text aligned on curves. OpenOffice.org also includes a printer setup tool with which you can select printers, fonts, paper sizes, and page formats.

NOTE *StarOffice is a fully integrated and Microsoft Office–compatible suite of office applications developed and supported by Sun Microsystems,* ***sun.com/staroffice****. Sun provides StarOffice as a commercial product, though educational use is free.*

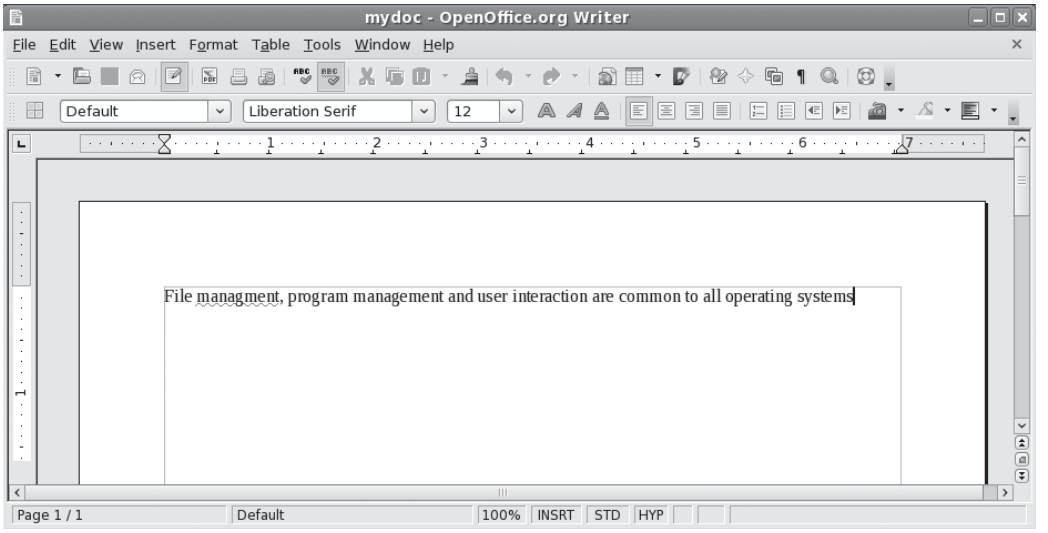


FIGURE 11-1 OpenOffice.org's Writer word processor

OpenOffice.org also provides access to many database files. File types supported include ODBC 3 (Open Database Connectivity), JDBC (Java), ADO, MySQL, dBase, Concurrent Versions System (CVS), PostgreSQL, and MDB (Microsoft Access) database files. Check the OpenOffice.org—Base page and Project page (**dba.openoffice.org**) for detailed information on drivers and supported databases.

OpenOffice.org features an underlying component model that can be programmed to develop customized applications. Check the OpenOffice.org API project for more details (**api.openoffice.org**). The OpenOffice.org Software Development Kit (SDK) provides support for using OpenOffice.org components in applications written in C++ or Java. The Unified Network Objects (UNO) model is the component model for OpenOffice.org, providing interaction between programming languages, other object models, and network connections.

KOffice

KOffice is an integrated office suite for the KDE (K Desktop Environment) consisting of several office applications, including a word processor, a spreadsheet, and graphics applications. You can download it using Pirut/Yum (Add/Remove Software). All applications are written for the KOM component model, which allows components from any one application to be used in another. This means you can embed a spreadsheet from KSpread or diagrams from Karbon14 in a KWord document. You can obtain more information about KOffice from the KOffice website at **koffice.org**.

Tip *KOffice applications have import and export filters that allow them to import or export files from popular applications like AbiWord, OpenOffice.org applications, MS Word, and even Palm documents. The reliability of these filters varies, and you should check the KOffice Filters web page for a listing of the different filters and their stability.*

KOffice Applications

Currently, KOffice includes KSpread, KPresenter, KWord, Karbon14, KFormula, KChart, Kugar, Krita, and Kivio (see Table 11-3). The contact application, Kontact, has been spun off as a separate project. Kontact is an integrated contact application including KMail, KOrganizer, KAddressbook, and KNotes. KSpread is a spreadsheet, KPresenter is a presentation application, Karbon14 is a vector graphics program, KWord is a Publisher-like word processor, KFormula is a formula editor, and KChart generates charts and diagrams. Kugar is a report generator, Krita is a bitmap image editor, and Kivio creates flow charts. Kexi provides database integration with KOffice applications, currently supporting PostgreSQL and MySQL.

KSpread, the spreadsheet application, incorporates the basic operations found in most spreadsheets, with formulas similar to those used in Excel. You can also embed charts, pictures, or formulas using KChart, Krita, Karbon14, or KFormula.

With KChart, you can create different kinds of charts, such as bar graphs, pie charts, and line graphs, as well as create diagrams. To generate a chart, you can use data in KSpread to enter your data. With KPresenter, you can create presentations consisting of text and graphics modeled using different fonts, orientations, and attributes such as colors. You can

Application	Description
KSpread	Spreadsheet
KPresenter	Presentation program
KOShell	Koffice Workspace for KOffice applications.
Karbon14	Vector graphics program
KWord	Word processor (desktop publisher)
KFormula	Mathematical formula editor
KChart	Tool for drawing charts and diagrams
Kugar	Report generator
Krita	Paint and image manipulation program
Kivio	Flow chart generator and editor (similar to Vivio)
Kexi	Database integration
KPlato	Project management and planning
Kontact (separate project)	Contact application including mail, address book, and organizer

Table 11-3 KOffice Applications

add such elements as speech bubbles, arrows, and clip art, as well as embed any KOffice component. Karbon14 is a vector-based graphics program, much like Adobe Illustrator and OpenOffice.org Draw. It supports the standard graphics operations such as rotating, scaling, and aligning objects.

KWord can best be described as a desktop publisher, with many of the features found in publishing applications like Microsoft Publisher and FrameMaker. Although it is also a fully functional word processor, KWord is not page-based like Word or WordPerfect. Instead, text is set up in frames that are placed on the page like objects. Frames, like objects in a drawing program, can be moved, resized, and even reoriented. You can organize frames into a frame set, having text flow from one to the other.

KParts

Embedded components support real-time updates. For example, if you use KChart to generate a chart in a KWord document using data in a KSpread spreadsheet and then change the selected data in the spreadsheet, KChart automatically updates the chart in the KWord document. In effect, you are creating a compound document—one made up of several applications. This capability is implemented by the KDE component model known as KParts. KParts provides communication between distributed objects. In this respect, you can think of an application working also as a server, providing other applications with the services it specializes in. A word processor, specializing in services such as paragraph formatting or spell-checking, could provide these services to all KOffice applications. In that way, other applications do not need to have their own text formatting functions written into them.

KParts is implemented with DCOP, the Desktop Communications Protocol. This is a very simple, small, and fast IPC/RPC mechanism for interprocess communication (IPC) that is based on the X Window System's ICE (Inter-Client Exchange) protocol. KDE applications

now use DCOP libraries to manage their communications with each other. DCOP makes development of KOffice applications much easier and more stable.

GNOME Office

The GNOME Office project supports three office applications: AbiWord, Gnumeric, and GNOME-DB. Former members of GNOME Office still provide certain Office tasks, like Novell's Evolution email and contact client. Many former members are still GNOME projects, with information listed for them at **gnome.org/projects**. You can find out more from the GNOME Office site at **gnome.org/gnome-office**. A current listing for common GNOME Office applications, including those not part of the GNOME Office suite, is shown in Table 11-4. All implement the CORBA model for embedding components, ensuring drag-and-drop capability throughout the GNOME interface.

Gnumeric, one of the GNOME Office applications, is a GNOME spreadsheet, a professional-level program meant to replace commercial spreadsheets. Like GNOME, Gnumeric is freely available under the GNU Public License. Gnumeric is included with the GNOME release, and you will find it installed with GNOME on any distribution that supports GNOME. You can download current versions from **gnome.org/projects/gnumeric**. Gnumeric supports standard GUI spreadsheet features, including autofilling and cell formatting, and it provides an extensive number of formats. It supports drag-and-drop operations, enabling you to select and then move or copy cells to another location. Gnumeric also supports plug-ins, making it possible to extend and customize its capabilities easily.

AbiWord, another GNOME Office application, is an open source word processor that aims to be a complete cross-platform solution, running on Mac, Unix, and Windows, as well as Linux. It is part of a set of desktop productivity applications being developed by the AbiSource project (**abisource.com**).

Application	Description
GNOME Office	
AbiWord	Cross-platform word processor
Gnumeric	Spreadsheet
GNOME-DB	Database connectivity
Other Office Apps for GNOME	
Evolution	Integrated email, calendar, and personal organizer (Novell)
Dia	Diagram and flow chart editor (GNOME project)
GnuCash	Personal finance manager (GNOME project)
Balsa	E-mail client (GNOME project)
Planner	Project manager (GNOME project)
OpenOffice.org	OpenOffice.org office suite

Table 11-4 GNOME Office and Other Office Applications for GNOME

The GNOME-DB project provides a GNOME Data Access (GDA) library supporting several kinds of databases, such as PostgreSQL, MySQL, Microsoft Access, and unixODBC. It provides an API to which databases can plug in. These back-end connections are based on CORBA. Through this API, GNOME applications can access a database. You can find out more about GNOME-DB at **gnome-db.org**.

Dia is a drawing program designed to create diagrams (GNOME project). You can select different kinds of diagrams to create, such as database, circuit object, flow chart, and network diagrams. You can easily create elements, along with lines and arcs with different types of endpoints such as arrows or diamonds. Data can be saved in XML format, making it easily transportable to other applications.

GnuCash (**gnucash.org**) is a personal finance application for managing accounts, stocks, and expenses (GNOME project). It includes support for home banking with the OpenHBCI interface. OpenHBCI is the open source home banking computer interface (**openhbci.sourceforge.net**).

Document Viewers (PostScript, PDF, and DVI)

PostScript, PDF, and DVI viewers are more commonly used with Office applications (see Table 11-5). Evince and Ghostview can display both PostScript (**.ps**) and PDF (**.pdf**) files. Ghostview's X Window System front end is **gv**. KPDF and Xpdf are PDF viewers. KPDF includes many of the standard Adobe Reader features such as zoom, two-page display, and full-screen mode. Alternatively, you can download Acrobat Reader for Linux from Adobe to display PDF files. All these viewers also have the ability to print documents. To generate PDF documents, you can use Scribus desktop publisher (**scribus.net**), and to edit PDF documents you can use pdfedit.

Linux also features a professional-level typesetting tool, called TeX, commonly used to compose complex mathematical formulas. TeX generates a DVI document that can then be displayed by DVI viewers, of which there are several for Linux. DVI files generated by the TeX document application can be viewed by KDVI, which is a plug-in to the KViewShell tool. KViewShell can display and print any kind of document for which it has a plug-in.

Viewer	Description
Evince	Document viewer for PostScript and PDF files
KPDF	KDE tool for displaying PDF files
KGhostView	KDE interface for displaying PostScript and PDF files
xpdf	X Window System tool for displaying PDF files only
KDVI	KDE tool for displaying TeX DVI files (plug-in to KViewShell)
Acrobat Reader	Adobe PDF and PostScript display application
Gnome-gv	Gnome Ghostscript viewer

Table 11-5 PostScript, PDF, and DVI viewers

PDA Access

For many PDAs you can use the pilot tools to access your handheld, transferring information between it and your system. The **pilot-link** package holds the tools you use to access your PDA. Check **pilot-link.org** for detailed documentation and useful links. The tool names usually begin with "pilot"; for instance, **pilot-addresses** reads addresses from an address book. Other tools whose names begin with "read" allow you to convert Palm data for access by other applications; **read-expenses**, for instance, outputs expense data as standard text. One of the more useful tools is **pilot-xfer**, used to back up your Palm.

Instead of using command line commands directly, you can use the J-Pilot, KPilot, and GNOMEPilot applications to access your Palm PDA. To use your PDA on GNOME, you can use the gnome-pilot applet from your GNOME panel to configure your connection. In the gnome-pilot applet's Preferences windows (right-click Applet), the Conduits panel lets you enable several hotsync operations to perform automatically, including email, memos, and installing files. Click the Help button for a detailed manual.

J-Pilot provides a GUI that lets you perform basic tasks such as synchronizing address books and writing memos. KPilot is included with the **kpim** package installed as part of the KDE Desktop. When you start up `kpilot` it will first let you automatically sync with your PDA. You then have the option to use either Evolution or KContact with your PDA, or just perform backups. You can then perform operations like hotsyncs, viewing addresses, and installing files. For text and Palm format conversions, you can use KPalmDoc. This tool will convert text files to Palm files, and Palm files to text files.

TIP *The device name used for your PDA is **/dev/pilot**, which is managed by **udev**. Should you need to manually specify a port for your handheld, you have to modify **udev** rules, not change the **/dev/pilot** file directly.*

Database Management Systems

Database software can be generally organized into three categories: SQL, Xbase, and desktop databases. *SQL-based databases* are professional-level relational databases whose files are managed by a central database server program. Applications that use the database do not access the files directly. Instead, they send requests to the database server, which then performs the actual access. *SQL* is the query language used on these industrial-strength databases. Both are open source projects freely available for your use. Table 11-6 lists DBMSs currently available for Linux.

The *Xbase language* is an enhanced version of the dBase programming language used to access database files whose formats were originally developed for dBase on the PC. With Xbase, DBMSs can directly access the database files. Xbase is used mainly for smaller personal databases, with database files often located on a user's own system.

SQL Databases (RDMS)

SQL databases are relational database management systems (RDMSs) designed for extensive database management tasks. Many of the major SQL databases now have Linux versions, including Oracle, Informix, Sybase, and IBM (but not, of course, Microsoft). These are

System	Site
PostgreSQL	The PostgreSQL database: **postgresql.org**
MySQL	MySQL database: **mysql.com**
Oracle	Oracle database: **oracle.com**
Sybase	Sybase database: **sybase.com**
DB2	IBM database: **software.ibm.com/data/db2/linux**
Informix	Informix database: **informix.com/linux**
MaxDB	SAP database now supported by MySQL: **mysql.com**
GNU SQL	The GNU SQL database: **ispras.ru/~kml/gss**
Flagship	Interface for Xbase database files: **fship.com/free.html**
Xbase	Xbase tools and libraries: **linux.techass.com/projects/xdb**

Table 11-6 Database Management Systems for Linux

commercial and professional database management systems of the highest order. Linux has proved itself capable of supporting complex and demanding database management tasks. In addition, many free SQL databases are available for Linux that offer much the same functionality. Most commercial databases also provide free personal versions, as do Oracle, Adabas D, and MySQL.

PostgreSQL

PostgreSQL is based on the POSTGRES DBMS, though it uses SQL as its query language. POSTGRES is a next-generation research prototype developed at the University of California, Berkeley. Linux versions of PostgreSQL are included in most distributions. You can find more information on it from the PostgreSQL website at **postgresql.org**. PostgreSQL is an open source project, developed under the GPL license.

MySQL

MySQL is a true multiuser, multithreaded SQL database server, supported by MySQL AB. MySQL is an open source product available free under the GPL license. You can obtain current information on it from its website, **mysql.com**. The site includes detailed documentation, including manuals and FAQs.

Oracle

Oracle offers a fully functional version of its Oracle9*i* DBMS for Linux, as well as the Oracle Application Server. You can download trial versions from the Oracle website at **oracle.com**. Oracle is a professional DBMS for large databases specifically designed for Internet e-business tasks. The Oracle Application Server provides support for real-time and commerce applications on the Web. As Linux is a fully functional version of Unix, Oracle is particularly effective on it. Oracle was originally designed to operate on Unix, and Linux is a far better platform for it than other PC operating systems.

Oracle offers extensive documentation for its Linux version that you can download from its Documentation page, to which you can link from the Support pages on its website.

The documentation available includes an installation guide, an administrator's reference, and release notes, as well as the generic documentation. You can find specific information on installing and configuring Oracle for Linux in the Oracle Database HOWTO.

Informix

Informix (now controlled by IBM) offers an integrated platform of Internet-based applications called Informix Internet Foundation.2000 on Linux. These include the Informix Dynamic Server, their database server. Informix Dynamic Server features Dynamic Scalable Architecture, making it capable of effectively using any hardware setup. Informix provides only commercial products. No free versions exist, though the company currently provides special promotions for Linux products. You can find out more about Informix at **www-4.ibm.com/software/data/informix**. Informix strongly supports Linux development of its Informix line. You can find out more about Informix for Linux at **www-306.ibm.com/software/data/informix/linux**.

Sybase

For Linux, Sybase offers the Sybase Adaptive Server Enterprise server (see **sybase.com**). You can currently download the Adaptive Server Enterprise server from the website. The Sybase Enterprise database features data integration that coordinates all information resources on a network. SQL Anywhere is a database system designed for smaller databases, though with the same level of complexity found in larger databases.

DB2

IBM provides a Linux version of its DB2 Universal Database software. You can download it free from the IBM DB2 web page for Linux, **software.ibm.com/data/db2/linux**. DB2 Universal Database for Linux includes Internet functionality along with support for Java and Perl. With the Web Control Center, administrators can maintain databases from a web browser. DB2 features scalability to expand the database easily, support for Binary Large Objects, and cost-based optimization for fast access. DB2 is still very much a mainframe database, though IBM is currently working on refining its Unix/Linux version.

MaxDB

MaxDB is a SAP-certified database, originally developed by SAP. It provides capabilities comparable to many of the professional-level databases. MaxDB is now developed by the MySQL AB project, **mysql.com**. Recently, the MySQL AB project also added MAX DB, formerly SAP DB.

GNU SQL

GNU SQL is the GNU relational database developed by a group at the Institute for System Programming of the Russian Academy of Sciences and supported by the GNU organization. It is a portable multiuser DBMS with a client/server structure that supports SQL. The server processes requests and performs basic administrative operations, such as unloading parts of the database used infrequently. The clients can reside on any computer of a local network. GNU SQL uses a dialect of SQL based on the SQL-89 standard and is designed for use on a Unix-like environment. You can download the database software from the GNU FTP site at **ftp.gnu.org**. For more information, contact the GNU SQL website at **ispras.ru/~kml/gss**.

Xbase Databases

Databases accessed with Xbase are smaller in scale, designed for small networks or for personal use. Many are originally PC database programs, such as dBase III, Clipper, FoxPro, and Quicksilver. Currently, only Flagship provides an interface for accessing Xbase database files.

Flagship is a compiler with which you can create interfaces for querying Xbase database files. The interfaces support menus and dialog boxes, and they have function calls that execute certain database queries. Flagship can compile dBase III+ code and up. It is compatible with dBase and Clipper, and can access most Xbase file formats, such as **.dbf**, **.dbt**, **.fmt**, and **.frm**. One of Flagship's key features is that its interfaces can be attached to a web page, enabling users to update databases. Flagship is commercial software, though you can download a free personal version from its website at **fship.com/free.html**.

Editors

Traditionally, most Linux distributions install the cursor-based editors Vim and Emacs. *Vim* is an enhanced version of the Vi text editor used on the Unix system. These editors use simple, cursor-based operations to give you a full-screen format. You can start these editors from the shell command line without any kind of X Window System support. In this mode, their cursor-based operations do not have the ease of use normally found in window-based editors. There are no menus, scroll bars, or mouse-click features. However, the K Desktop and GNOME do support powerful GUI text editors with all these features. These editors operate much more like those found on Macintosh and Windows systems. They have full mouse support, scroll bars, and menus. You may find them much easier to use than the Vi and Emacs editors. These editors operate from their respective desktops, requiring you first have either KDE or GNOME installed, though the editors can run on either desktop. Vi and Emacs have powerful editing features that have been refined over the years. Emacs, in particular, is extensible to a full-development environment for programming new applications. Newer versions of Emacs, such as GNU Emacs and XEmacs, provide X Window System support with mouse, menu, and window operations. They can run on any window manager or desktop. In addition, the gvim version of the Vim editor also provides basic window operations. You can access it on both GNOME and KDE desktops. Table 11-7 lists several GUI-based editors for Linux.

GNOME Editor: Gedit

The Gedit editor is a basic text editor for the GNOME desktop. It provides full mouse support, implementing standard GUI operations, such as cut and paste to move text, and click and drag to select text. It supports standard text editing operations such as Find and Replace. You can use Gedit to create and modify your text files, including configuration files. Gedit also provides more advanced features such as print preview and configurable levels of undo/redo operations, and it can read data from pipes. It features a plug-in menu that provides added functionality, and it includes plug-ins for spell-checking, encryption, email, and text-based web page display.

K Desktop Editors: Kate, KEdit, and KJots

All the K Desktop editors provide full mouse support, implementing standard GUI operations, such as cut and paste to move text and click and drag to select text. Kate is an advanced editor, with such features as spell-checking, font selection, and highlighting. Most commands can be

The K Desktop	Description
KEdit	Text editor
Kate	Text and program editor
KJots	Notebook editor
KWord	Desktop publisher, part of KOffice
GNOME	
Gedit	Text editor
AbiWord	Word processor
X Window System	
GNU Emacs	Emacs editor with X Window System support
XEmacs	X Window System version of Emacs editor
gvim	Vim version with X Window System support (vim-x11)
OpenWriter	OpenOffice.org word processor that can edit text files

TABLE 11-7 Linux Desktop Editors

selected using menus. A toolbar of icons for common operations is displayed across the top of the Kate window. A sidebar displays panels for a file selector and a file list. With the file selector, you can navigate through the file system selecting files to work on. Kate also supports multiple views of a document, letting you display segments in their own windows, vertically or horizontally. You can also open several documents at the same time, moving between them with the file list. Kate is designed to be a program editor for editing software programming/development-related source code files. Although Kate does not have all the features of Emacs or Vi, it can handle most major tasks. Kate can format the syntax for different programming languages, such as C, Perl, Java, and XML. In addition, Kate has the capability to access and edit files on an FTP site or website.

KEdit is an older, simple text editor meant for editing simple text files such as configuration files. A toolbar of buttons at the top of the KEdit window enables you to execute common editing commands easily using just a mouse click. With KEdit, you can also mail files you are editing over a network. The entry for KEdit in the K menu is listed simply as Text Editor.

The editor KJots is designed to enable you to jot down notes in a notebook. It organizes notes you write into notebooks, called simply *books*. You can select the one you want to view or add to from the Books menu. To start KJots, select its entry in the Utilities | Pim menu or enter the `kjots` command in a terminal window.

The Emacs Editor

Emacs can best be described as a working environment featuring an editor, a mailer, a newsreader, and a Lisp interpreter. The editor is tailored for program development, enabling you to format source code according to the programming language you use. Many versions of Emacs are currently available for use on Unix and Linux systems. The versions usually

included with Linux distributions are either GNU Emacs or XEmacs. The current version for GNU Emacs is 20.*x*; it is X Window System–capable, enabling GUI features such as menus, scroll bars, and mouse-based editing operations. Check the update FTP sites for your distribution for new versions as they come out, as well as the GNU website at **gnu.org** and the Emacs website at **emacs.org**. You can also find out more information about XEmacs at its website, **xemacs.org**.

Emacs derives much of its power and flexibility from its capability to manipulate buffers. Emacs can be described as a buffer-oriented editor. Whenever you edit a file in any editor, the file is copied into a work buffer, and editing operations are done in the work buffer. Emacs can manage many work buffers at once, enabling you to edit several files at the same time. You can edit buffers that hold deleted or copied text. You can even create buffers of your own, fill them with text, and later save them to a file. Emacs extends the concept of buffers to cover any task. When you compose mail, you open a mail buffer; when you read news, you open a news buffer. Switching from one task to another is simply a matter of switching to another buffer.

The Emacs editor operates much like a standard word processor. The keys on your keyboard represent input characters. Commands are implemented with special keys, such as control (CTRL) keys and alternate (ALT) keys. There is no special input mode, as in Vi or Ed. You type in your text, and if you need to execute an editing command, such as moving the cursor or saving text, you use a CTRL key. Such an organization makes the Emacs editor easy to use. However, Emacs is anything but simple—it is a sophisticated and flexible editor with several hundred commands. Emacs also has special features, such as multiple windows. You can display two windows for text at the same time. You can also open and work on more than one file at a time, displaying each on the screen in its own window. You invoke the Emacs editor with the command **emacs**. You can enter the name of the file you want to edit, and if the file does not exist, it is created. In the next example, the user prepares to edit the file **mydata** with Emacs:

```
$ emacs mydata
```

The GNU Emacs editor now supports an X Window System graphical user interface. To enable X support, start Emacs within an X Window System environment, such as a KDE, GNOME, or XFce desktop. The basic GUI editing operations are supported: selection of text with click-and-drag mouse operations; cut, copy, and paste; and a scroll bar for moving through text. The Mode line and Echo areas are displayed at the bottom of the window, where you can enter keyboard commands. The scroll bar is located on the left side. To move the scroll bar down, click it with the left mouse button. To move the scroll bar up, click it with the right mouse button.

NOTE *XEmacs is the complete Emacs editor with a graphical user interface and Internet applications, including a web browser, a mail utility, and a newsreader.*

The Vi Editor: Vim and Gvim

The Vim editor included with most Linux distributions is an enhanced version of the Vi editor. It includes all the commands and features of the Vi editor. Vi, which stands for *visual*, remains one of the most widely used editors in Linux. Keyboard-based editors like Vim and

Emacs use a keyboard for two different operations: to specify editing commands and to receive character input. Used for editing commands, certain keys perform deletions, some execute changes, and others perform cursor movement. Used for character input, keys represent characters that can be entered into the file being edited. Usually, these two different functions are divided among different keys on the keyboard. Alphabetic keys are reserved for character input, while function keys and control keys specify editing commands, such as deleting text or moving the cursor. Such editors can rely on the existence of an extended keyboard that includes function and control keys. Editors in Unix, however, were designed to assume a minimal keyboard with alphanumeric characters and some control characters, as well as the ESC and ENTER keys. Instead of dividing the command and input functions among different keys, the Vi editor has three separate modes of operation for the keyboard: command and input modes, and a line editing mode. In *command* mode, all the keys on the keyboard become editing commands; in the *input* mode, the keys on the keyboard become input characters. Some of the editing commands, such as **a** or **i**, enter the input mode. On typing **i**, you leave the command mode and enter the input mode. Each key then represents a character to be input to the text. Pressing ESC automatically returns you to the command mode, and the keys once again become editor commands. As you edit text, you are constantly moving from the command mode to the input mode and back again. With Vim, you can use the CTRL-O command to jump quickly to the command mode and enter a command, and then automatically return to the input mode. Table 11-8 lists a very basic set of Vi commands to get you started.

Command	Cursor Movement
h	Moves the cursor left one character.
l	Moves the cursor right one character.
k	Moves the cursor up one line.
j	Moves the cursor down one line.
CTRL-F	Moves forward by a screen of text; the next screen of text is displayed.
CTRL-B	Moves backward by a screen of text; the previous screen of text is displayed.
Input	*All input commands place the user in input; the user leaves input with ESC.*
a	Enters input after the cursor.
i	Enters input before the cursor.
o	Enters input below the line the cursor is on; inserts a new empty line below the one the cursor is currently on.
Text Selection (Vim)	**Cursor Movement**
v	Visual mode; move the cursor to expand selected text by character. Once selected, press key to execute action: **c** change, **d** delete, **y** copy, **:** line editing command, **J** join lines, **U** uppercase, **u** lowercase.

TABLE 11-8 Vi Editor Commands

Text Selection (Vim)	Cursor Movement
`V`	Visual mode; move cursor to expand selected text by line.
Delete	**Effect**
`x`	Deletes the character the cursor is on.
`dd`	Deletes the line the cursor is on.
Change	*(Except for the replace command, `r`, all change commands place the user into input after deleting text.)*
`cw`	Deletes the word the cursor is on and places the user into the input mode.
`r`	Replaces the character the cursor is on. After pressing `r`, the user enters the replacement character. The change is made without entering input; the user remains in the Vi command mode.
`R`	First places into input mode, and then overwrites character by character. Appears as an overwrite mode on the screen but actually is in input mode.
Move	*Moves text by first deleting it, moving the cursor to desired place of insertion, and then pressing the* `p` *command. (When text is deleted, it is automatically held in a special buffer.)*
`p`	Inserts deleted or copied text after the character or line the cursor is on.
`P`	Inserts deleted or copied text before the character or line the cursor is on.
`dw p`	Deletes a word, and then moves it to the place you indicate with the cursor (press `p` to insert the word *after* the word the cursor is on).
`yy` or `Y p`	Copies the line the cursor is on.
Search	*The two search commands open up a line at the bottom of the screen and enable the user to enter a pattern to be searched for; press* ENTER *after typing in the pattern.*
/pattern	Searches forward in the text for a pattern.
?pattern	Searches backward in the text for a pattern.
`n`	Repeats the previous search, whether it was forward or backward.
Line Editing Commands	**Effect**
`w`	Saves file.
`q`	Quits editor; `q!` quits without saving.

TABLE 11-8 Vi Editor Commands (*continued*)

Although the Vi command mode handles most editing operations, it cannot perform some, such as file saving and global substitutions. For such operations, you need to execute line editing commands. You enter the line editing mode using the Vi colon command. The colon is a special command that enables you to perform a one-line editing operation. When you type the colon, a line opens up at the bottom of the screen with the cursor placed at the beginning of the line. You are now in the line editing mode. In this mode, you enter an editing

command on a line, press ENTER, and the command is executed. Entry into this mode is usually only temporary. Upon pressing ENTER, you are automatically returned to the Vi command mode, and the cursor returns to its previous position on the screen.

Although you can create, save, close, and quit files with the Vi editor, the commands for each are not all that similar. Saving and quitting a file involves the use of special line editing commands, whereas closing a file is a Vi editing command. Creation of a file is usually specified on the same shell command line that invokes the Vi editor. To edit a file, type **vi** or **vim** and the name of a file on the shell command line. If a file by that name does not exist, the system creates it. In effect, giving the name of a file that does not yet exist instructs the Vi editor to create that file. The following command invokes the Vi editor, working on the file **booklist**. If **booklist** does not yet exist, the Vi editor creates it.

```
$ vim booklist
```

After executing the **vim** command, you enter Vi's command mode. Each key becomes a Vi editing command, and the screen becomes a window onto the text file. Text is displayed screen by screen. The first screen of text is displayed, and the cursor is positioned in the upper-left corner. With a newly created file, there is no text to display. This fact is indicated by a column of tildes at the left side of the screen. The tildes represent the part of a screen that is not part of the file.

Remember, when you first enter the Vi editor, you are in the command mode. To enter text, you need to enter the input mode. In the command mode, **a** is the editor command for appending text. Pressing this key places you in the input mode. Now the keyboard operates like a typewriter and you can input text to the file. If you press ENTER, you merely start a new line of text. With Vim, you can use the arrow keys to move from one part of the entered text to another and work on different parts of the text. After entering text, you can leave the input mode and return to the command mode by pressing ESC. Once finished with the editing session, you exit Vi by typing two capital *Z*s, **ZZ**. Hold down the SHIFT key and press **Z** twice. This sequence first saves the file and then exits the Vi editor, returning you to the Linux shell. To save a file while editing, you use the line editing command **w**, which writes a file to the disk; **w** is equivalent to the Save command found in other word processors. You first type a colon to access the line editing mode, and then type **w** and press ENTER.

You can use the **:q** command to quit an editing session. Unlike the **ZZ** command, the **:q** command does not perform any save operation before it quits. In this respect, it has one major constraint. If any modifications have been made to your file since the last save operation, the **:q** command will fail and you will not leave the editor. However, you can override this restriction by placing a **!** qualifier after the **:q** command. The command **:q!** will quit the Vi editor without saving any modifications made to the file in that session since the last save (the combination **:wq** is the same as **ZZ**).

To obtain online help, enter the **:help** command. This is a line editing command. Type a colon, enter the word **help** on the line that opens at the bottom of the screen, and then press ENTER. You can add the name of a specific command after the word **help**. The F1 key also brings up online help.

As an alternative to using Vim in a command line interface, you can use gvim, which provides X Window System–based menus for basic file, editing, and window operations. Gvim is installed as the **vim-x11** package, which includes several links to gvim such as **evim**, **gview**, and **gex** (open Ex editor line). To use gvim, you can select it from your distribution's main menu, or enter the **gvim** command at an X Window System terminal prompt.

The standard Vi interface is displayed, but with several menu buttons displayed across the top along with a toolbar with buttons, for common commands like searches and file saves. All the standard Vi commands work just as described previously. However, you can use your mouse to select items on these menus. You can open and close a file, or open several files using split windows or different windows. The editing menu enables you to cut, copy, and paste text as well as undo or redo operations. In the editing mode, you can select text with your mouse with a click-and-drag operation, or use the Editing menu to cut or copy and then paste the selected text. Text entry, however, is still performed using the `a`, `i`, or `o` command to enter the input mode. Searches and replacements are supported through a dialog window. There are also buttons on the toolbar for finding next and previous instances. Gview also features programming support, with color coding for programming syntax, for both shell scripts and C++ programs. There is even a Make button for running Makefiles.

CHAPTER 12

Graphics Tools and Multimedia

Linux supports a wide range of both graphics and multimedia applications and tools, such as simple image viewers like KView, sophisticated image manipulation programs like GIMP, music and CD players like Rhythmbox, and TV viewers like Totem. Graphics tools available for use under Linux are listed later in this chapter in Table 12-2. Additionally, there is strong support for multimedia tasks, from video and DVD to sound and music editing (see Table 12-3, also later). Thousands of multimedia and graphics projects, as well as standard projects, are under development or currently available from online and distribution repositories like **sourceforge.net**, **freshmeat.net**, or Fedora's **freshrpms.net** (see Table 12-1). Be sure to check the **sourceforge.net** site for any kind of application you may need.

NOTE *Support for many popular multimedia operations, specifically MP3, DVD, and DivX, are not included with many distributions, including Fedora and Red Hat, because of licensing and other restrictions. To play MP3, DVD, or DivX files, you will have to download and install support packages manually. For Fedora, precompiled Red Hat Package Manager (RPM) binary packages for many popular media applications and libraries, such as MPlayer and XviD as well as MP3 and DVD video support, are available at* ***rpm.livna.org*** *and* ***freshrpms.net.***

Graphics Tools

GNOME, KDE, and the X Window System support an impressive number of graphics tools, including image viewers, window grabbers, image editors, and paint tools. On the KDE and GNOME desktops, these tools can be found under either a Graphics submenu or the Utilities menu.

NOTE *Linux has become a platform of choice for many professional-level multimedia tasks such as generating computer-generated images (CGIs) and animation for movie special effects, using such demanding software as Maya and Softimage. Linux graphics libraries include those for OpenGL, MESA, and SGI.*

Project	Description and Site
SourceForge.net	This site holds a massive amount of multimedia software for Linux, much under development: **sourceforge.net**
KDE multimedia applications	KDE supports an extensive set of multimedia software applications: **kde-apps.org**
GNOME multimedia applications	Many multimedia applications have been developed for GNOME: **gnomefiles.org**
Sound & MIDI Software for Linux	This site lists a wide range of multimedia and sound software. **linux-sound.org**
Advanced Linux Sound Architecture (ALSA)	The (ALSA) project is under development on Linux under the GPL: **alsa-project.org**
Open Sound System	Open Sound System has a wide range of supporting multimedia applications: **opensound.com**

Table 12-1 Linux Multimedia Projects and Sites

Photo Management Tools: F-Spot and digiKam

The F-Spot Photo Manager provides a simple and powerful way to manage, display, and import your photos and images (**f-spot.org**). Photos can be organized by different categories such as events, people, and places. You can perform standard display operations like rotation or full-screen viewing, along with slide shows. Image editing support is provided. Selected photos can be directly burned to a CD (using Nautilus burning capabilities).

Features include a simple and easy-to-use interface. A timeline feature lets you see photos as they were taken. You can also display photos in full-screen mode or as slide shows.

F-Spot includes a photo editor that provides basic adjustments and changes like rotation, red-eye correction, and standard color settings including temperature and saturation.

You can tag photos and place them in groups, making them easier to access. With a tag, you can label a collection of photos, then use the tag to instantly access them. The tag itself can be a user-selected icon, including one that the user can create with the Tag icon editor.

F-Spot provides several ways to upload photos to a website. It provides direct access to a Flickr account (**flickr.com**) or to Gallery-supported sites (**gallery.menalto.com**). Photos can also be saved to a folder for later uploading a website, either as plain files or as static HTML files.

digiKam is a KDE photo manager with many of the same features (**digiKam.org**). A side panel allows easy access by album, date, tags, or previous searches. digiKam also provides image editing capabilities, with numerous effects. The digiKam configuration (Settings menu) provides extensive options including image editing, digital camera support, and interface configuration.

KDE Graphics Tools

KView is a simple image viewer for GIF and JPEG image files. The KSnapshot program is a simple screen grabber for KDE, which currently supports only a few image formats. KFourier is an image-processing tool that uses the Fourier transform to apply several filters to an image at once. KuickShow is an easy-to-use, comfortable image browser and viewer supporting slide shows and numerous image formats based on imlib. KolourPaint is a simple paint program with brushes, shapes, and color effects; it supports numerous image formats. Krita is the KOffice professional image paint and editing application, with a wide range of features such as creating web images and modifying photographs (formerly known as Krayon and KImageShop).

GNOME Graphics Tools

GNOME features several powerful and easy-to-use graphics tools. Some are installed with Linux, whereas you can download others, such as GView and gtKam, from **gnomefiles.org**. Also, many of the KDE tools work just as effectively in GNOME and are accessible from the GNOME desktop.

The gThumb application is a thumbnail image viewer that lets you browse and display images using thumbnails and organize them into catalogs for easy reference. See **sourceforge.net** for more information.

GIMP is the GNU Image Manipulation Program, a sophisticated image application much like Adobe Photoshop. You can use GIMP for such tasks as photo retouching, image composition, and image authoring. It supports features such as layers, channels, blends, and gradients. GIMP makes particular use of the GTK+ widget set. You can find out more about GIMP and download the newest versions from its website at **gimp.org**. GIMP is freely distributed under the GNU Public License.

Inkscape is a GNOME-based vector graphics application for SVG (scalable vector graphics) images. It features capabilities similar to professional-level vector graphics applications like Adobe Illustrator. The SVG format allows easy generation of images for web use as well as complex art. Though its native format is SVG, it can also export to PNG format. It features layers and easy object creation, including stars and spirals. A color bar lets you quickly change color fills.

The gPhoto project provides software for accessing digital cameras (**gphoto.org**). Several front-end interfaces are provided for a core library, called **libgphoto2**, consisting of drivers and tools that can access numerous digital cameras.

X Window System Graphics Programs

X Window System–based applications run directly on the underlying X Window System, which supports the more complex desktops like GNOME and KDE. These applications tend to be simpler, lacking the desktop functionality found in GNOME or KDE applications. Xpaint is a paint program much like MacPaint that allows you to load graphics or photographs and then create shapes, add text and colors, and use brush tools with various sizes and colors. Xfig is a drawing program, and Xmorph enables you to morph images, changing their shapes. ImageMagick lets you convert images from one format to another; you can, for instance, change a TIFF image to a JPEG image. Table 12-2 lists some popular graphics tools for Linux.

Tools	Description
Photo Management	
F-Spot	GNOME digital camera application and image library manager (**f-spot.org**)
digiKam	KDE digital camera application and image library manager (**digikam.org**)
KView	Simple image viewer for GIF and JPEG image files
ShowFoto	Simple image viewer, works with digiKam (**digikam.org**)
KSnapshot	Screen grabber
KFourier	Image processing tool that uses the Fourier transform
KuickShow	Image browser and viewer
KolourPaint	Paint program
Krita	Image editor (**koffice.org/krita**)
GNOME	
gThumb	Image browser, viewer, and cataloger (**gthumb.sourceforge.net**)
GIMP	GNU Image Manipulation Program (**gimp.org**)
Inkscape	GNOME Vector graphics application (**inkscape.org**)
X Window System	
Xpaint	Paint program
Xfig	Drawing program
Xmorph	Tool that morphs images
Xfractals	Fractal image generator
ImageMagick	Image format conversion and editing tool

TABLE 12-2 Graphics Tools for Linux

Multimedia

Many applications are available for both video and sound, including sound editors, MP3 players, and video players (see Table 12-3). Linux sound applications include mixers, digital audio tools, CD audio writers, MP3 players, and network audio support. There are literally thousands of projects currently under development at **sourceforge.net**. If you are looking for a specific kind of application, odds are you will find it there. Current projects include a full-featured video player, a digital video recorder, and a digital audio mixer. Many applications designed specifically for the GNOME or KDE user interface can be found at their respective software sites (**gnomefiles.org** and **kde-apps.org**). Precompiled binary RPM or Debian package manager (DEB) packages can be easily downloaded and installed from distribution repositories.

Multimedia applications use various codecs to run different kinds of media, such as MP3 for music files. The Codec Buddy tool will detect the codec you need and download it if not installed. You can purchase third-party commercial codecs like Windows Media or Dolby codecs from Fluendo (**fluendo.com**).

Application	Description
Xine	Multimedia player for video, DVD, and audio
Rhythmbox	Music management (GStreamer)
Sound Juicer	GNOME CD audio ripper (GStreamer)
Grip	CD audio ripper
aKtion	KDE video player
Kscd	Music CD player
Krec	KDE sound recorder
Kaboodle	A media player
GNOME CD Player	CD player
GNOME Sound Recorder	Sound recorder
Pulse	Pulse sound server
XMMS	CD player
Xplaycd	Music CD player
Noatun	KDE multimedia player
Xanim	Animation and video player
RealPlayer	RealMedia and RealAudio streaming media (**real.com**)
HelixPlayer	Open source version of RealPlayer (**real.com**)
K3b	KDE CD writing interface for cdrecord, mkisofs, and cdda2wav
KAudioCreator	KDE CD burner and ripper
dvdauthor	Tools for creating DVDs (**dvdauthor.sourceforge.net**).
Qauthor	KDE front end for dvdauthor (**kde-apps.org**)
DVDStyler	DVD authoring application for GNOME (**dvdstyler.sourceforge.net**).
Fluendo	Commercial multimedia codecs for Linux (**fluendo.com**)
Codec Buddy	Codec Buddy tool

TABLE 12-3 Multimedia and Sound Applications

GStreamer

Many of the GNOME-based applications use GStreamer, which is a streaming media framework based on graphs and filters. Using a plug-in structure, GStreamer applications can accommodate a wide variety of media types. You can download modules and plug-ins from **gstreamer.freedesktop.org**. GNOME on Linux includes several GStreamer applications:

- The Totem video player uses GStreamer to play DVDs, VCDs, and MPEG media.
- Rhythmbox provides integrated music management; it is similar to the Apple iTunes music player.

- Sound Juicer is an audio CD ripper.
- A CD player, a sound recorder, and a volume control are all provided as part of the GStreamer GNOME Media package.

Multimedia System Selector

GStreamer can be configured to use different input and output sound and video drivers and servers. You can make these selections using the GStreamer properties tool. To open this tool from the Desktop menu, first select Preferences, then More Preferences, and then the Multimedia Systems Selector entry. You can also enter **`gstreamer-properties`** in a terminal window. The properties window displays two tabbed panels, one for sound and the other for video. The output drivers and servers are labeled Default Sink, and the input divers are labeled Default Source. There are pop-up menus for each, listing the available sound or video drivers or servers. For example, the sound server used is ALSA, but you can change that to OSS.

Gstreamer Plug-ins: The Good, the Bad, and the Ugly

Many GNOME multimedia applications like Totem use Gstreamer to provide multimedia support. To use such features as DVD Video and MP3, you have to install Gstreamer extra plug-ins. You can find out more information about Gstreamer and its supporting packages at **gstreamer.freedesktop.org**.

The supporting packages can be confusing. For version 1 and above, Gstreamer establishes four different support packages called the base, the good, the bad, and the ugly. The base package is a set of useful plug-ins that are reliable. The good package is a set of supported and tested plug-ins that meet all licensing requirements. The bad is a set of unsupported plug-ins whose performance is not guaranteed and may crash, but still meet licensing requirements. The ugly package contains plug-ins that work fine but may not meet licensing requirements, such as DVD support.

- **The base** Reliable commonly used plug-ins
- **The good** Reliable additional and useful plug-ins
- **The ugly** Reliable but not fully licensed plug-ins (DVD/MP3 support)
- **The bad** Possibly unreliable but useful plug-ins (possible crashes)

GStreamer MP3 Compatibility: iPod

For your iPod and other MP3 devices to work with GNOME applications like Rhythmbox, you will need to install MP3 support for GStreamer. MP3 support is not included with several distributions because of licensing issues. You can, however, download and install the GStreamer **gstreamer-plugins-ugly** support package as noted previously, which maintains most multimedia support packages that are not included with most Linux distributions.

To sync and import from your iPod, you can use iPod management software such as GUI for iPod (gtkpod). Several scripts and tools are currently available for iPod operations; they include SyncPOD, myPod, gtkpod (a GUI for iPod), and iPod for Linux. Check **sourceforge.net** and search for iPod.

Sound Applications

Sound devices on Linux are supported by drivers, forming a sound system. With the current kernel, sound support is implemented by the Advanced Linux Sound Architecture

(ALSA) system. ALSA replaces the free version of the Open Sound System used in previous releases, as well as the original built-in sound drivers. You can find more about ALSA at **alsa-project.org**.

MP3 with LAME

LAME originally stood for "Lame Ain't an Mp3 Encoder," but it has long since evolved into a full MP3 encoder whose software is available under the LPGL license. It is included with VideoLAN and FFmpeg, and it will download in support of MPlayer or Xine.

Because of licensing and patent issues, many Linux distributions have removed support for MP3 files. MP3 playback capability has been removed from multimedia players like XMMS and Noatun. As an alternative to MP3, you can use Ogg Vorbis compression for music files (**vorbis.com**).

Music Applications

Many music applications are currently available for GNOME, including sound editors, MP3 players, and audio players. You can use the GNOME CD Player to play music CDs and the GNOME Sound Recorder to record sound sources. Check the software map at **gnomefiles .org** for current releases. A variety of applications are also available for KDE, including two media players (Kaiman and Kaboodle), a mixer (KMix), and a CD player (Kscd). Check **kde-apps.org** for recent additions. Several X Window System–based multimedia applications are installed with most distributions. These include XMMS and Xplaycd, CD music players, and Xanim, an animation and video player.

GNOME includes the XMMS multimedia player, the GNOME CD Player, the GNOME Sound Recorder, and the GNOME Volume Control in the Sound And Video menu. KDE applications include KMidi, Kaboodle, and Noatun. Linux systems also support HelixPlayer, the open source project used for RealPlayer. HelixPlayer runs only open source media like Ogg Vorbis files (though you can obtain RealPlayer audio and video codecs for the player). See **helixcommunity.org** for more information. You can also download a copy of RealPlayer, the Internet streaming media player, from **real.com**. Be sure to choose RealPlayer for Unix.

The Sound & Midi Software for Linux site **(linux-sound.org)** holds links to websites and FTP sites for many of sound applications.

The Pulse sound server lets you direct and manage sound streams from a devices, letting you direct and modify sound to different clients.

CD Burners and Rippers

Several CD writer programs that can be used for CD music and MP3 writing (burners and rippers) are available from **kde-apps.org**. These include K3b, CD-Rchive, and KAudioCreator (CD ripper). For GNOME, you can use CD-REC and the Nautilus CD burner, which is integrated into the Nautilus file manager, the default file manager for the GNOME desktop. All use mkisofs, cdrecord, and cdda2wav CD-writing programs, which are installed as part of your distribution. GNOME also features two CD audio rippers, Grip and Sound Juicer.

Tip *If your CD or DVD application has difficulty finding your CD/DVD player or burner, you may need to check whether HAL is creating an appropriate link to your CD/DVD device using* ***/dev/cdrom*** *or* ***/dev/dvdrom****. These links should be generated automatically.*

Video Applications

Several projects are under way to provide TV, video, DivX, DVD, and DTV support for Linux (see Table 12-4). In most cases, the most recent versions will be in source code format on the original site. For these you will have to download the source code, which you will then need to compile and install. In effect, Firefox provides nearly seamless install or extraction operations with downloads. For RPM packages, Firefox will give you the option to automatically install the RPM with system-install-packages. For compressed archives such as **.tar.bz** files, Firefox will automatically invoke File Roller, letting you immediately decompress and extract source code files to a selected directory.

Video and DVD Players

Access to current DVD and media players is provided at **dvd.sourceforge.net**. Here you will find links for players like VideoLAN, MPlayer, and Xine.

- The VideoLAN project (**videolan.org**) offers network streaming support for most media formats, including MPEG-4 and MPEG-2. It includes a multimedia player, VLC, that can work on any kind of system.

Projects and Players	Descriptions and Sites
LinuxTV.org	Links to video, TV, and DVD sites: **linuxtv.org**
DVD players list	**dvd.sourceforge.net**
xine	Xine video player: **xinehq.de**
Totem	Totem video and DVD player for GNOME, based on Xine and using GStreamer: **xinehq.de**
VideoLAN	Network multimedia streaming, includes x264 high-definition support: **videolan.org**
MPlayer	MPlayer DVD/multimedia player: **mplayerhq.hu**
PowerDVD	Cyberlink PowerDVD for Linux **gocyberlink.com**
DVD::rip	DVD transcoding and DivX software: **exit1.org/dvdrip**
kdetv	KDE TV viewer
tvtime	TV viewer: **tvtime.sourceforge.net**
DivX for Linux	labs.divx.com/DivXLinuxCodec
XviD	Open Source DivX, may be included with distributions: **xvid.org**

Table 12-4 Video and DVD Projects and Applications

- MPlayer is one of the most popular and capable multimedia/DVD players in use. It is a cross-platform open source alternative to RealPlayer and Windows Media Player, and it includes support for DivX. You can download MPlayer from **mplayerhq.hu**. MPlayer uses an extensive set of supporting libraries and applications like **lirc**, **lame**, **lzo**, and **aalib**, which are also on the site. If you have trouble displaying video, be sure to check the preferences for different video devices and select one that works best.
- Xine is a multipurpose video player for Linux/Unix systems that can play video, DVD, and audio discs. See **xinehq.de** for more information.
- Totem is a GNOME movie player based on Xine that uses GStreamer. To expand Totem capabilities, you need to install added GStreamer plug-ins, such as the DivX plug-in to display DivX files.
- For DVD transcoding and DivX support, check the DVD::rip project (**exit1.org/dvdrip)**.
- VideoLAN is another popular player requiring a list of supporting packages.
- Additional codec support is supplied by ffmeg and x264. The x264 codec is an open source version of the high-definition H.264 codec developed by VideoLAN.

None of the open source software hosted at SourceForge.net performs CSS decryption of commercial DVDs. You can, however, download and install the **libdvdcss** library, which works around CSS decryption by treating the DVD as a block device, allowing you to use any of the DVD players to run commercial DVDs. It also provides region-free access. Bear in mind that this may be not be legal in certain countries that require CSS licenses for DVD players.

Originally, many of these players did not support DVD menus. With the **libdvdnav** library, these players now feature full DVD menu support. The **libdvdread** library provides basic DVD interface support, such as reading IFO files.

TV Players

The site **linuxtv.org** provides detailed links to DVD, digital video broadcasting (DVB), and multicasting. The site also provides downloads of many Linux video applications.

tvtime is a TV player, which works with many common video-capture cards, relying on drivers developed for TV tuner chips on those cards, like the Conexant chips. It can only display a TV image. It has no recording or file playback capabilities. Check **tvtime .sourceforge.net** for more information.

For KDE, several video applications are available or currently under development, including kdetv. Check **kde-apps.org** for downloads. For GNOME players, check **gnomefiles.org**.

DVB and HDTV Support

For DVB and HDTV reception, you can use most DVB cards, as well as many HDTV cards like the PCHDTV video card (**pdhdtv.org**). For example, the latest PCHDTV card uses the **cx88-dvb** drivers included with most recent Linux kernel (for earlier kernels versions you would have to download, compile, and install a separate driver). The DVB kernel driver may not be installed by default. In this case you would have to use modprobe to manually install it (on Debian, place the module name in /etc/modules to have it loaded automatically). You can use the lsmod command to see if your DVB module is loaded.

Many DVB-capable applications like Kaffeine already have DVB accessibility installed. You can also use the dvb-tools to manage access; they include **scan** for scanning your

channels and **zap** tools for accessing the signal directly. You would first need to create a channels configuration file using the scanning tool. For the PC-HDTV card you can use the dvb-atsc-tools which you can download and install from their website, www.pchdtv.com. The tools are in source code and you can use a simple make and make install commands to create and install them (be sure kernel headers and software development support are installed).The tools may work with any conexant compliant HDTV cards like Fusion HDTV.

The DVB tools can also be used to record HDTV and DVB broadcasts to TS (transport stream) files.

The **transport stream (.ts or .tp)** file can then be viewed with an HDTV capable viewer, such as the HDTV versions of Xine or the Videolan VLC media player. You can use MythTV and Xine to view and record. Check the MythTV site for details (**mythtv.org**). Be sure appropriate decoders are installed like mpeg2, FFmpeg, and A52 (ac3). For DVB broadcasts, many DVB capable players and tools like Kaffeine and Klear, as well as vdr will tune and record DVB broadcasts in t, s, and c formats. The dvb-tools package holds sample configurations.

DivX and Xvid on Linux

DivX is a commercial video compression technology (free for personal use) for providing DVD-quality video with relatively small file sizes. You can compress 60 minutes of DVD video into about 400MB while maintaining very good quality. DivX is based on the MPEG-4 compression format, whereas DVD is MPEG-2. You can download the Linux version of DivX for free from **labs.divx.com/DivXLinuxCodec**. You have to manually install the package. If you download with Firefox, you can choose to extract the archive directly.

Instead of trying to get DivX to work, you can just use the open source version of DivX known as Xvid. Most DivX files can be run using XviD. XviD is an entirely independent open source project, but it's compatible with DivX files. Most distributions provide easily installed software packages for Xvid. You can also download the XviD source code from **xvid.org**.

13 CHAPTER

Mail and News Clients

Your Linux system supports a wide range of both electronic mail and news clients. Mail clients enable you to send and receive messages with other users on your system or accessible from your network. News clients let you read articles and messages posted in a newsgroups, which are open to access by all users. This chapter reviews mail and news clients installed with Linux.

Mail Clients

You can send and receive email messages in a variety of ways, depending on the type of mail client you use. Although all electronic mail utilities perform the same basic tasks of receiving and sending messages, they tend to have different interfaces. Some mail clients operate on a desktop, such as KDE or GNOME. Others run on any X Window System manager. Several popular mail clients were designed to use a screen-based interface and can be started only from the command line. Other traditional mail clients were developed for just the command line interface, which requires you to type your commands on a single command line. Most mail clients described here are included in standard Linux distributions and come in a standard package for easy installation. For web-based Internet mail services, such as Hotmail, Google, and Yahoo, you use a web browser instead of a mail client to access mail accounts provided by those services. Table 13-1 lists several popular Linux mail clients. Mail is transported to and from destinations using mail transport agents. Sendmail, Exim, and Smail send and receive mail from destinations on the Internet or at other sites on a network. To send mail over the Internet, they use the Simple Mail Transport Protocol (SMTP). Most Linux distributions automatically install and locally configure Sendmail for you. On starting up your system, having configured your network connections, you can send and receive messages over the Internet.

You can sign your email message with the same standard signature information, such as your name, Internet address or addresses, or farewell phrase. Having your signature information automatically added to your messages is helpful. To do so, you need to create a signature file in your home directory and enter your signature information in it. A *signature file* is a standard text file you can edit using any text editor. Mail clients such as KMail enable you to specify a file to function as your signature file. Others, such as Mail, expect the signature file to be named **.signature**.

Mail Client	Description
Kontact (KMail, KAddressbook, KOrganizer)	Includes the K Desktop mail client, KMail; integrated mail, address book, and scheduler
Evolution	Email client
Balsa	GNOME mail client
Thunderbird	Mozilla group standalone mail client and newsreader
Netscape	Web browser–based mail client
GNUEmacs and XEmacs	Emacs mail clients
Mutt	Screen-based mail client
Sylpheed	Gtk mail and news client
Mail	Original Unix-based command line mail client
SquirrelMail	Web-based mail client

TABLE 13-1 Linux Mail Clients

MIME

MIME (Multipurpose Internet Mail Extensions) is used to enable mail clients to send and receive multimedia files and files using different character sets such as those for different languages. Multimedia files can be images, sound clips, or even video. Mail clients that support MIME can send binary files automatically as attachments to messages. MIME-capable mail clients maintain a file called **mailcap** that maps different types of MIME messages to applications on your system that can view or display them. For example, an image file will be mapped to an application that can display images. Your mail clients can then run that program to display the image message. A sound file will be mapped to an application that can play sound files on your speakers. Most mail clients have MIME capabilities built in and use their own version of the **mailcap** file. Others use a program called metamail that adds MIME support. MIME is not only used in mail clients; both the KDE and GNOME file managers use MIME to map a file to a particular application so that you can launch the application directly from the file.

The mime.types File

Applications are associated with binary files by means of the **mailcap** and **mime.types** files. The **mime.types** file defines different MIME types, associating a MIME type with a certain application. The **mailcap** file then associates each MIME type with a specified application. Your system maintains its own MIME types file, usually **/etc/mime.types**.

Entries in the MIME types file associate a MIME type and possible subtype of an application with a set of possible file extensions used for files that run on a given kind of application. The MIME type is usually further qualified by a subtype, separated from the major type by a slash. For example, a MIME type image can have several subtypes, such as JPEG, GIF, or TIFF. A sample MIME type entry defining a MIME type for JPEG files is shown here. The MIME type is **image/jpeg**, and the list of possible file extensions is "**jpeg jpg jpe**":

```
image/jpeg jpeg jpg jpe
```

The applications specified will depend on those available on your particular system. The MIME type is separated from the application with a semicolon. In many cases, X Window System–based programs are specified. Comments are indicated with a **#**. A ***** used in a MIME subtype references all subtypes. The entry **image/*** would be used for an application that could run all types of image files. A formatting code, **%s**, is used to reference the attachment file that will be run on this application. Sample **mailcap** entries are shown here. The first entry associates all **image** files with the xv image viewer. The next two associate video and video MPEG files with the XAnim application.

```
image/*; xv %s
video/*; xanim %s
video/mpeg; xanim %s
```

MIME Associations on GNOME and KDE

You can also create and edit MIME types on the GNOME and KDE desktops. For GNOME, use the GNOME Control Center's MIME types capplet. This capplet will list the MIME types defined for your system along with their associated filename extensions. Edit an entry to change the application and icon associated with that MIME type, that type of file. On KDE, use the KDE Control Center's File Association entry under KDE Components. This will list MIME types and their associated filename extensions. Select an entry to edit it and change the applications associated with it. KDE saves its MIME type information in a separate file called **mimelnk** in the KDE configuration directory.

MIME Standard Associations

Though you can create your own MIME types, a standard set is already in use. The types text, image, audio, video, application, multipart, and message, along with their subtypes, have already been defined for your system. You will find that commonly used file extensions such as **.tif** and **.jpg** for TIFF and JPEG image files are already associated with a MIME type and an application. Though you can easily change the associated application, it is best to keep the MIME types already installed. The current official MIME types are listed at the IANA website (**iana.org**) under the name Media Types, provided as part of their Assignment Services. You can access the media types file directly on their site.

OpenPGP/MIME and S/MIME Authentication and Encryption Protocols

S/MIME and OpenPGP/MIME are authentication protocols for signing and encrypting mail messages. S/MIME was originally developed by the RSA Data Security. OpenPGP is an open standard based on the PGP/MIME protocol developed by the PGP (Pretty Good Privacy) group. Clients like KMail and Evolution can use OpenPGP/MIME to authenticate messages. Check the Internet Mail Consortium for more information, **imc.org**.

Evolution

Evolution is the primary mail client for the GNOME desktop. It is installed by default along with OpenOffice. Though designed for GNOME, it will work equally well on KDE. Evolution is an integrated mail client, calendar, and address book currently being developed by Novell and now known as the Novell Evolution. The Evolution mailer is a powerful tool with support for numerous protocols (SMTP, POP, and IMAP), multiple mail accounts, and encryption. With Evolution, you can create multiple mail accounts on different servers,

including those that use different protocols such as POP or IMAP. You can also decrypt PGP- or GPG-encrypted messages.

The Evolution mailer provides a simple GUI, with a toolbar for commonly used commands and a sidebar for shortcuts. A menu of Evolution commands allows access to other operations. The main panel is divided into two panes, one for listing the mail headers and the other for displaying the currently selected message. You can click any header title to sort your headers by that category. Evolution also supports the use of virtual folders. These are folders created by the user to hold mail that meets specified criteria. Incoming mail can be automatically distributed to their particular virtual folder. For automatic mail notification, you use the mail-notification plug-in for Evolution.

Thunderbird

Thunderbird is a full-featured standalone email client provided by the Mozilla project (**mozilla.org**). It is designed to be easy to use, highly customized, and heavily secure. It features advanced intelligent spam filtering, as well as security features like encryption, digital signatures, and S/MIME. To protect against viruses, email attachments can be examined without being run. Thunderbird supports both IMAP and POP, as well as the use of LDAP address books. It functions as a newsreader and features a built-in RSS reader. In addition, Thunderbird is an extensible application, allowing customized modules to be added to enhance its capabilities. You can download extensions such as dictionary search and contact sidebars from its website. GPG encryption can be supported with the **enigmail** extension.

The interface uses a standard three-pane format, with a side pane for listing mail accounts and their boxes. The top pane lists main entries, and the bottom pane shows text. Commands can be run using the toolbar, menus, or keyboard shortcuts. You can even change the appearance using different themes. Thunderbird also supports HTML mail, displaying web components like URLs in mail messages.

The message list pane will show several fields by which you can sort your messages. Some use symbols like the Threads, Attachments, and Read icons. Clicking Threads will gather the messages into respective threads with replies grouped together. The last icon in the message list fields is a pop-up menu letting you choose which fields to display. Thunderbird provides a variety of customizable display filters, such as People I Know, which displays only messages from those in your address book, and Attachments, which displays messages with attached files. You can even create your own display filters. Search and sorting capabilities also include filters that can match selected patterns in any field, including subject, date, or the message body.

When you first start up Thunderbird, you will be prompted to create an email account. You can add more email accounts or modify your current ones by selecting Account Settings from the Edit menu. Then click Add Account to open a dialog with four options, one of which is an email account. Upon selecting the Email option, you are prompted to enter your email address and name. In the next panel you specify either the POP or IMAP protocol and enter the name of the incoming email server, such as **smtp.myemailserver.com**. You then specify an incoming username given you by your email service. Then you enter an account name label to identify the account on Thunderbird. A final verification screen lets you confirm your entries. In the Account Settings window you will see an entry for your news server, with panels for Server Settings, Copies & Folders, Composition & Addressing,

Offline & Disk Space, Return Receipt, and Security. The Server Settings panel has entries for your server name, port, username, and connection and task configurations such as automatically downloading new messages. The Security panel opens the Certificate Manager, where you can select security certificates to digitally sign or encrypt messages.

Thunderbird provides an address book where you can enter complete contact information, including email addresses, street addresses, phone numbers, and notes. Select Address Book from the Tools menu to open the Address Book window. There are three panes, one for the address books available, one listing the address entries with field entries such as name, email, and organization, and one for displaying address information. You can sort the entries by these fields. Clicking an entry will display the address information, including email address, street addresses, and phone. Only fields with values are displayed. To create a new entry in an address book, click New Card to open a window with panels for Contact and Address information. To create mailing lists from the address book entries, click the New List button, specify the name of the list, and enter the email addresses.

Once you have your address book set up, you can use its addresses when creating mail messages easily. On the Compose window, click the Contacts button to open a Contacts pane. Your address book entries will be listed using the contact's name. Just click the name to add it to the address box of your email message. Alternatively, you can open the address book and drag and drop addresses to an address box on your message window.

A user's email messages, addresses, and configuration information are kept in files located in the **.thunderbird** directory within the user's home directory. Backing up this information is as simple as making a copy of that directory. Messages for the different mail boxes are kept in a **Mail** subdirectory. If you are migrating to a new system, you can just copy the directory from the older system. To back up the mail for any given mail account, copy the **Mail** subdirectory for that account. Though the default address books, **abook.mab** and **history.mab**, can be interchangeably copied, nondefault address books need to be exported to an LDIF format and then imported to the new Thunderbird application. It is advisable to regularly export your address books to LDIF files as backups.

GNOME Mail Clients: Evolution, Balsa, and Others

Several GNOME-based mail clients are now available (see Table 13-2). These include Evolution, Balsa, and Sylpheed (Evolution is included with GNOME). Check **gnomefiles.org** for more mail clients as they come out. Many are based on the GNOME mail client libraries (camel), which provides support for standard mail operations.

Application	Description
Balsa	E-mail client for GNOME that supports POP3, IMAP, local folders, and multithreading
Evolution	Integrated mail client, calendar, and contact manager
Sylpheed	Mail and news client similar to Windows clients
gnubiff	E-mail checker and notification tool
Mail Notification	E-mail checker and notification that works with numerous mail clients, including MH, Sylpheed, Gmail, Evolution, and Mail

Table 13-2 GNOME Mail Clients

Balsa provides a full-featured GUI for composing, sending, and receiving mail messages. The Balsa window displays three panes for folders, headers, and messages. The left pane displays your mail folders. You initially have three folders: an inbox folder for received mail, an outbox folder for mail you have composed but have not sent yet, and a trash folder for messages you have deleted. You can also create your own mail folders in which you can store particular messages. To place a message in a folder you have created, click and drag the message header for that message to the folder.

The K Desktop Mail Client: KMail

The K Desktop mail client, KMail, provides a full-featured GUI for composing, sending, and receiving mail messages. KMail is now part of the KDE Personal Information Management suite, KDE-PIM, which also includes an address book (KAddressbook), an organizer and scheduler (KOrganizer), and a note writer (KNotes). All these components are also directly integrated on the desktop into Kontact. To start up KMail, you start the Kontact application. The KMail window displays three panes for folders, headers, and messages. The upper-left pane displays your mail folders. You have an inbox folder for received mail, an outbox folder for mail you have composed but have not sent yet, and a sent-mail folder for messages you have previously sent. You can create your own mail folders and save selected messages in them, if you wish. The top-right pane displays mail headers for the currently selected mail folder. To display a message, click its header. The message is then displayed in the large pane below the header list. You can also send and receive attachments, including binary files. Pictures and movies that are received are displayed using the appropriate K Desktop utility. If you right-click the message, a pop-up menu displays options for actions you may want to perform on it. You can move or copy it to another folder or simply delete it. You can also compose a reply or forward the message. KMail, along with Kontact, KOrganizer, and KAddressbook, is accessible from the KDE Desktop, Office, and Internet menus.

To set up KMail for use with your mail accounts, you must enter account information. Select the Configure entry in the Settings menu. Several panels are available on the Settings window, which you can display by clicking their icons in the left column. For accounts, you select the Network panel. You may have more than one mail account on mail servers maintained by your ISP or LAN. A configure window is displayed where you can enter login, password, and host information. For secure access, KMail now supports SSL, provided OpenSSL is installed. Messages can now be encrypted and decoded by users. It also supports IMAP in addition to POP and SMTP protocols.

SquirrelMail Web Mail Client

You can use the SquirrelMail web mail tool to access mail on a Linux system using your web browser. It will display a login screen for mail users. It features an inbox list and message reader, support for editing and sending new messages, and a plug-in structure for adding new features. You can find out more about SquirrelMail at **squirrelmail.org**. The Apache configuration file is **/etc/httpd/conf.d/squirrelmail.conf**, and SquirrelMail is installed in **/usr/share/squirrelmail**. Be sure that the IMAP mail server is also installed.

To configure SquirrelMail, you use the **config.pl** script in the **/usr/share/squirrelmail/config** directory. This displays a simple text-based menu where you can configure settings like the server to use, folder defaults, general options, and organizational preferences.

```
./config.pl
```

To access SquirrelMail, use the web server address with the **/squirrelmail** extension, as in **locahost/squirrelmail** for users on the local system or **mytrek.com/squirrelmail** for remote users.

Emacs

The Emacs mail clients are integrated into the Emacs environment, of which the Emacs editor is the primary application. They are, however, fully functional mail clients. The GNU version of Emacs includes a mail client along with other components, such as a newsreader and editor. GNU Emacs is included in Linux distributions. Check the Emacs website at **gnu.org/software/emacs** for more information. When you start up GNU Emacs, menu buttons are displayed across the top of the screen. If you are running Emacs in an X Window System environment, you have full GUI capabilities and can select menus using your mouse. To access the Emacs mail client, select from the mail entries in the Tools menu. To compose and send messages, just select the Send Mail item in the Tools menu. This opens a screen with prompts for To and Subject header entries. You then type the message below, using any of the Emacs editing capabilities. GNU Emacs is a working environment within which you can perform a variety of tasks, with each task having its own buffer. When you read mail, a buffer opens to hold the header list, and when you read a message, another buffer holds the contents. When you compose a message, yet another buffer holds the text you wrote. The buffers you have opened for mail, news, or editing notes or files are listed in the Buffers menu. You can use this menu to switch among them.

XEmacs is another version of Emacs, designed to operate solely with a GUI. The Internet applications, which you can easily access from the main XEmacs button bar, include a web browser, a mail utility, and a newsreader. When composing a message, you have full use of the Emacs editor with all its features, including the spell-checker and search/replace.

Command Line Mail Clients

Several mail clients use a simple command line interface. They can be run without any other kind of support, such as the X Window System, desktops, or cursor support. They are simple and easy to use but include an extensive set of features and options. Two of the more widely used mail clients of this type are Mail and Mutt. Mail is the mailx mail client that was developed for the Unix system. It is considered a kind of default mail client that can be found on all Unix and Linux systems. Mutt is a cursor-based client that can be run from the command line.

NOTE *You can also use the Emacs mail client from the command line, as described in the previous section.*

Mutt

Mutt has an easy-to-use screen-based interface and an extensive set of features, such as MIME support. You can find more information about Mutt from the Mutt website, **mutt.org**. Here you can download recent versions of Mutt and access online manuals and help resources. On most distributions, the Mutt manual is located in the **/usr/doc** directory under Mutt. The Mutt newsgroup is **comp.mail.mutt**, where you can post queries and discuss recent Mutt developments.

Mail

What is known now as the Mail utility was originally created for BSD Unix and called simply mail. Later versions of Unix System V adopted the BSD mail utility and renamed it mailx; now it is referred to as Mail. Mail functions as a de facto default mail client on Unix and Linux systems. All systems have the mail client called Mail, whereas they may not have other mail clients.Check the **mail** Man page for detailed information and commands.

To send a message with Mail, type **`mail`** along with the address of the person to whom you are sending the message. Press ENTER and you are prompted for a subject. Enter the subject of the message and press ENTER again. At this point, you are placed in input mode. Anything typed in is taken as the contents of the message. Pressing ENTER adds a new line to the text. When you finish typing your message, press CTRL-D on a line of its own to end the message. You will then be prompted to enter a user to whom to send a carbon copy of the message (Cc). If you do not want to sent a carbon copy, just press ENTER. You will then see *EOT (end-of-transmission)* displayed after you press CTRL-D to end your message.

You can send a message to several users at the same time by listing those users' addresses as arguments on the command line following the **`mail`** command. In the next example, the user sends the same message to both **chris** and **aleina**.

```
$ mail chris aleina
```

To receive mail, you enter only the **`mail`** command and press ENTER. This invokes a Mail shell with its own prompt and mail commands. A list of message headers is displayed. Header information is arranged into fields beginning with the status of the message and the message number. The status of a message is indicated by a single uppercase letter, usually **`N`** for *new* or **`U`** for *unread.* A message number, used for easy reference to your messages, follows the status field. The next field is the address of the sender, followed by the date and time the message was received and then the number of lines and characters in the message. The last field contains the subject the sender gave for the message. After the headers, the Mail shell displays its prompt, an ampersand (**&**). At the Mail prompt, you enter commands that operate on the messages. An example of a Mail header and prompt follows:

```
$ mail
Mail version 8.1 6/6/93. Type ? for help.
"/var/spool/mail/larisa": 3 messages 2 unread
 1 chris@turtle.mytrek. Thu Jun 7 14:17 22/554 "trip"
>U 2 aleina@turtle.mytrek Thu Jun 7 14:18 22/525 "party"
 U 3 dylan@turtle.mytrek. Thu Jun 7 14:18 22/528 "newsletter"
& q
```

Mail references messages either through a message list or through the current message marker (**>**). The greater-than sign (**>**) is placed before a message considered the current message. The current message is referenced by default when no message number is included with a Mail command. You can also reference messages using a message list consisting of several message numbers. Given the messages in the preceding example, you can reference all three messages with **`1-3`**.

You use the **`R`** and **`r`** commands to reply to a message you have received. The **`R`** command entered with a message number generates a header for sending a message and then places you into the input mode to type in the message. The **`q`** command quits Mail. When you quit,

messages you have already read are placed in a file called **mbox** in your home directory. Instead of saving messages in the **mbox** file, you can use the **s** command to save a message explicitly to a file of your choice. Mail has its own initialization file, called **.mailrc**, that is executed each time Mail is invoked, for either sending or receiving messages. Within it, you can define Mail options and create Mail aliases. You can set options that add different features to Mail, such as changing the prompt or saving copies of messages you send. To define an alias, you enter the keyword **alias**, followed by the alias you have chosen and then the list of addresses it represents. In the next example, the alias **myclass** is defined in the **.mailrc** file.

```
.mailrc
alias myclass chris dylan aleina justin larisa
```

In the next example, the contents of the file **homework** are sent to all the users whose addresses are aliased by **myclass**.

```
$ mail myclass < homework
```

Notifications of Received Mail

As your mail messages are received, they are automatically placed in your mailbox file, but you are not automatically notified when you receive a message. You can use a mail client to retrieve any new messages, or you can use a mail monitor tool to tell you if you have any mail waiting. Several mail notification tools are also available, such as gnubiff and Mail Notification. Mail Notification will support Gmail, as well as Evolution (for Evolution, install the separate plug-in package). When you first log in after Mail Notification has been installed, the Mail Notification configuration window is displayed. Here you can add new mail accounts to check, such as Gmail accounts, as well as set other features like summary pop-ups. When you receive mail, a Mail icon will appear in the notification applet of your panel. Move your cursor over it to check for any new mail. Clicking it will display the Mail Notification configuration window, though you can configure this to go directly to your email application. gnubiff will notify you of any POP3 or IMAP mail arrivals.

The KDE Desktop has a mail monitor utility called Korn that works in much the same way. Korn shows an empty inbox tray when there is no mail and a tray with slanted letters in it when mail arrives. If old mail is still in your mailbox, letters are displayed in a neat square. You can set these icons as any image you want. You can also specify the mail client to use and the polling interval for checking for new mail. If you have several mail accounts, you can set up a Korn profile for each one. Different icons can appear for each account, telling you when mail arrives in one of them.

For command line interfaces, you can use the biff utility. The biff utility notifies you immediately when a message is received. This is helpful when you are expecting a message and want to know as soon as it arrives. Then biff automatically displays the header and beginning lines of messages as they are received. To turn on biff, you enter **biff y** on the command line. To turn it off, you enter **biff n**. To find out if biff is turned on, enter **biff** alone.

You can temporarily block biff by using the **mesg n** command to prevent any message displays on your screen. The **mesg n** command not only stops any Write and Talk messages, it also stops biff and Notify messages. Later, you can unblock biff with a **mesg y** command. A **mesg n** command comes in handy if you don't want to be disturbed while working on some project.

Accessing Mail on Remote POP Mail Servers

Most newer mail clients are equipped to access mail accounts on remote servers. For such mail clients, you can specify a separate mail account with its own mailbox. For example, if you are using an ISP, most likely you will use that ISP's mail server to receive mail. You will have set up a mail account with a username and password for accessing your mail. Your email address is usually your username and the ISP's domain name. For example, a username of **justin** for an ISP domain named **mynet.com** would have the address **justin@mynet.com**. The address of the actual mail server could be something like **mail.mynet.com**. The user **justin** would log in to the **mail.mynet.com** server using the username **justin** and password to access mail sent to the address **justin@mynet.com**. Mail clients, such as Evolution, KMail, Balsa, and Thunderbird, enable you to set up a mailbox for such an account and access your ISP's mail server to check for and download received mail. You must specify what protocol a mail server uses. This is usually either the POP or the IMAP protocol (IMAP). This procedure is used for any remote mail server. Using a mail server address, you can access your account with your username and password.

TIP *Many mail clients, such as Mutt and Thunderbird, support IMAP and POP directly.*

Should you have several remote email accounts, instead of creating separate mailboxes for each in a mail client, you can arrange to have mail from those accounts sent directly to the inbox maintained by your Linux system for your Linux account. All your mail, whether from other users on your Linux system or from remote mail accounts, will appear in your local inbox. Such a feature is helpful if you are using a mail client, such as Mail, that does not have the capability to access mail on your ISP's mail server. You can implement such a feature with Fetchmail, which checks for mail on remote mail servers and downloads it to your local inbox, where it appears as newly received mail (you will have to be connected to the Internet or the remote mail server's network).

To use Fetchmail, you have to know a remote mail server's Internet address and mail protocol. Most remote mail servers use the POP3 protocol, but others may use the IMAP or POP2 protocols. Enter **`fetchmail`** on the command line with the mail server address and any needed options. The mail protocol is indicated with the **`-p`** option and the mail server type, usually POP3. If your email username is different from your Linux login name, you use the **`-u`** option and the email name. Once you execute the **`fetchmail`** command, you are prompted for a password. The syntax for the **`fetchmail`** command for a POP3 mail server follows:

```
fetchmail -p POP3 -u username mail-server
```

To use Fetchmail, connect to your ISP and then enter the **`fetchmail`** command with the options and the POP server name on the command line. You will see messages telling you if mail is there and, if so, how many messages are being downloaded. You can then use a mail client to read the messages from your inbox. You can run Fetchmail in daemon mode to have it automatically check for mail. You have to include an option specifying the interval in seconds for checking mail.

```
fetchmail -d 1200
```

You can specify options such as the server type, username, and password in a **.fetchmailrc** file in your home directory. You can also have entries for other mail servers and accounts you may have. Once it is configured, you can enter `fetchmail` with no arguments; it will read entries from your **.fetchmailrc** file. You can also make entries directly in the **.fetchmailrc** file. An entry in the **.fetchmailrc** file for a particular mail account consists of several fields and their values: poll, protocol, username, and password. *Poll* is used to specify the mail server name, and *protocol*, the type of protocol used. Notice you can also specify your password, instead of having to enter it each time Fetchmail accesses the mail server.

Mailing Lists

As an alternative to newsgroups, you can subscribe to mailing lists. Users on mailing lists automatically receive messages and articles sent to the lists. Mailing lists work much like a mail alias, broadcasting messages to all users on the list. Mailing lists were designed to serve small, specialized groups of people. Instead of posting articles for anyone to see, only those who subscribe receive them. Numerous mailing lists, as well as other subjects, are available for Linux. For example, at the **gnome.org** site, you can subscribe to any of several mailing lists on GNOME topics, such as **gnome-themes-list@gnome.org**, which deals with GNOME desktop themes. You can do the same at **lists.kde.org** for KDE topics. At **liszt.com**, you can search for mailing lists on various topics. By convention, to subscribe to a list, you send a request to the mailing list address with a `-request` term added to its username. For example, to subscribe to **gnome-themes-list@gnome.org**, you send a request to **gnome-themes-list-request@gnome.org**. At **linux.org**, you can link to sites that support Linux-oriented mailing lists, such as the Linux Mailing Lists website. Lists exist for such topics as the Linux kernel, administration, security, and different distributions. For example, **linux-admin** covers administration topics, and **linux-apps** discusses software applications; **vger.kernel.org** provides mailing list services for Linux kernel developers.

NOTE *You can use the Mailman and Majordomo programs to automatically manage your mailing lists. Mailman is the GNU mailing list manager (**list.org**). You can find out more about Majordomo at **greatcircle.com/majordomo** and about Mailman at **sourceforge.net**.*

Usenet News

Usenet is an open mail system on which users post messages that include news, discussions, and opinions. It operates like a mailbox that any user on your Linux system can read or send messages to. Users' messages are incorporated into Usenet files, which are distributed to any system signed up to receive them. Each system that receives Usenet files is referred to as a *site*. Certain sites perform organizational and distribution operations for Usenet, receiving messages from other sites and organizing them into Usenet files, which are then broadcast to many other sites. Such sites are called *backbone sites*, and they operate like publishers, receiving articles and organizing them into different groups.

To access Usenet news, you need access to a news server. A news server receives the daily Usenet newsfeeds and makes them accessible to other systems. Your network may have a system that operates as a news server. If you are using an ISP, a news server is probably maintained by your ISP for your use. To read Usenet articles, you use a *newsreader*—a client program that connects to a news server and accesses the articles. On the Internet and in TCP/IP networks, news servers communicate with newsreaders using the Network News Transfer Protocol (NNTP) and are often referred to as NNTP news servers. Or, you could also create your own news server on your Linux system to run a local Usenet news service or to download and maintain the full set of Usenet articles. Several Linux programs, called *news transport agents,* can be used to create such a server. This chapter focuses on the variety of newsreaders available for the Linux platform.

Usenet files were originally designed to function like journals. Messages contained in the files are referred to as *articles.* A user could write an article, post it in Usenet, and have it immediately distributed to other systems around the world. Someone could then read the article on Usenet, instead of waiting for a journal publication. Usenet files themselves were organized as journal publications. Because journals are designed to address specific groups, Usenet files were organized according to groups called *newsgroups.* When a user posts an article, it is assigned to a specific newsgroup. If another user wants to read that article, he or she looks at the articles in that newsgroup. You can think of each newsgroup as a constantly updated magazine. For example, to read articles on the Linux operating system, you access the Usenet newsgroup on Linux. Usenet files are also used as bulletin boards on which people carry on debates. Again, such files are classified into newsgroups, though their articles read more like conversations than journal articles. You can also create articles of your own, which you can then add to a newsgroup for others to read. Adding an article to a newsgroup is called *posting* the article.

NOTE *The Google website maintains online access to Usenet newsgroups. It has the added capability of letting you search extensive newsgroup archives. You can easily locate articles on similar topics that may reside in different newsgroups. Other sites such as Yahoo maintain their own groups that operate much like Usenet newsgroups, but with more supervision.*

Linux has newsgroups on various topics. Some are for discussion, and others are sources of information about recent developments. On some, you can ask for help for specific problems. A selection of some of the popular Linux newsgroups is provided here:

Newsgroup	Topic
comp.os.linux.announce	Announcements of Linux developments
comp.os.linux.admin	System administration questions
comp.os.linux.misc	Special questions and issues
comp.os.linux.setup	Installation problems
comp.os.linux.help	Questions and answers for particular problems
linux.help	Help for Linux problems

Newsreaders

You read Usenet articles with a newsreader, such as KNode, Pan, Mozilla, trn, or tin, which enables you to first select a specific newsgroup and then read the articles in it. A newsreader operates like a user interface, enabling you to browse through and select available articles for reading, saving, or printing. Most newsreaders employ a sophisticated retrieval feature called *threads* that pulls together articles on the same discussion or topic. Newsreaders are designed to operate using certain kinds of interfaces. For example, KNode is a KDE newsreader that has a KDE interface and is designed for the KDE desktop. Pan has a GNOME interface and is designed to operate on the GNOME desktop. Pine is a cursor-based newsreader, meaning that it provides a full-screen interface that you can work with using a simple screen-based cursor that you can move with arrow keys. It does not support a mouse or any other GUI feature. The tin program uses a simple command line interface with limited cursor support. Most commands you type in and press ENTER to execute. Several popular newsreaders are listed in Table 13-3.

NOTE *Numerous newsreaders currently are under development for both GNOME and KDE. You can check for KDE newsreaders on the software list on the K Desktop website at* ***kde-apps.org.*** *For GNOME newsreaders, check Internet tools on the software map on the GNOME website at* **gnome-files.org**.

Most newsreaders can read Usenet news provided on remote news servers that use the NNTP. Many such remote news servers are available through the Internet. Desktop newsreaders, such as KNode and Pan, have you specify the Internet address for the remote news server in their own configuration settings. Several shell-based newsreaders, however, such as trn and tin, obtain the news server's Internet address from the **NNTPSERVER** shell variable. Before you can connect to a remote news server with such newsreaders, you first have to assign the Internet address of the news server to the **NNTPSERVER** shell variable, and then export that variable. You can place the assignment and export of **NNTPSERVER** in a login initialization file, such as **.bash_profile**, so that it is performed automatically

Newsreader	Description
Pan	GNOME desktop newsreader
KNode	KDE desktop newsreader
Thunderbird	Mail client with newsreader capabilities (X based)
Sylpheed	GNOME Windows-like newsreader
slrn	Newsreader (cursor based)
Emacs	Emacs editor, mail client, and newsreader (cursor based)
trn	Newsreader (command line interface)
NewsBin	Newsreader (Windows version works under Wine)

TABLE 13-3 Linux Newsreaders

whenever you log in. Administrators can place this entry in the **/etc/profile** file for a news server available to all users on the system.

```
$ NNTPSERVER=news.domain.com
$ export NNTPSERVER
```

The **slrn** newsreader is screen based. Commands are displayed across the top of the screen and can be executed using the listed keys. Different types of screens exist for the newsgroup list, article list, and article content, each with its own set of commands. An initial screen lists your subscribed newsgroups with commands for posting, listing, and subscribing to your newsgroups. When you start **slrn** for the first time, you will have to create a **.jnewsrc** file in your home directory. Use the following command: `slrn -f .jnewsrc -create`. Also, you will have to set the `NNTPSERVER` variable and make sure it is exported.

The **slrn** newsreader features a new utility called **slrnpull** that you can use to automatically download articles in specified newsgroups. This allows you to view your selected newsgroups offline. The **slrnpull** utility was designed as a simple single-user version of Leafnode; it will access a news server and download its designated newsgroups, making them available through **slrn** whenever the user chooses to examine them. Newsgroup articles are downloaded to the **SLRNPULL_ROOT** directory, usually **/var/spool/srlnpull**. The selected newsgroups to be downloaded are entered in the **slrnpull.conf** configuration file placed in the **SLRNPULL_ROOT** directory. In this file, you can specify how many articles to download for each group and when they should expire. To use **slrn** with **slrnpull**, you will have to further configure the **.slrnrc** file to reference the **slrnpull** directories where newsgroup files are kept.

NOTE *Several Windows-based newsreaders, like the popular Newsbin, will run under Linux, using the Wine Windows emulation. To get the newsreader working, you will have to follow specific configuration directions, which often require specific Windows DLLs. For Newsbin, check the Newsbin forum for Linux. Be sure to add Windows DLLs to your* ***.wine/drive_c/Windows/System32*** *directory.*

News Transport Agents

Usenet news is provided over the Internet as a daily newsfeed of articles and postings for thousands of newsgroups. This newsfeed is sent to sites that can then provide access to the news for other systems through newsreaders. These sites operate as news servers; the newsreaders used to access them are their clients. The news server software, called *news transport agents,* is what provides newsreaders with news, enabling you to read newsgroups and post articles. For Linux, three of the popular news transport agents are INN, Leafnode, and Cnews. Both Cnews and Leafnode are small, simple, and useful for small networks. INN is more powerful and complex, designed with large systems in mind (see **isc.org** for more details).

Daily newsfeeds on Usenet are often large and consume much of a news server's resources in both time and memory. For this reason, you may not want to set up your own Linux system to receive such newsfeeds. If you are operating in a network of Linux systems, you can designate one of them as the news server and install the news transport agent on it to receive and manage the Usenet newsfeeds. Users on other systems on your network can then access that news server with their own newsreaders.

If your network already has a news server, you needn't install a news transport agent at all. You only have to use your newsreaders to remotely access that server (see **NNTPSERVER** in the preceding section). In the case of an ISP, such providers often operate their own news servers, which you can also remotely access using your own newsreaders, such as KNode and Pan. Remember, though, that newsreaders must take the time to download the articles for selected newsgroups, as well as updated information on all the newsgroups.

You can also use news transport agents to run local versions of news for only the users on your system or your local network. To do this, install INN, Leafnode, **slrnpull**, or Cnews and configure them just to manage local newsgroups. Users on your system will then be able to post articles and read local news.

This page intentionally left blank

CHAPTER 14

Web, FTP, and Java Clients

Most Linux distributions will provide powerful web and FTP clients for accessing the Internet. Many are installed automatically and are ready to use when you first start up your Linux system. Linux also includes full Java development support, letting you run and construct Java applets. This chapter will cover some of the more popular web, Java, and FTP clients available on Linux. Web and FTP clients connect to sites that run servers, using web pages and FTP files to provide services to users.

Web Clients

The World Wide Web (WWW, or the Web) is a hypertext database of different types of information, distributed across many different sites on the Internet. A *hypertext database* consists of items linked to other items, which, in turn, may be linked to yet other items, and so on. Upon retrieving an item, you can use that item to retrieve any related items. For example, you can retrieve an article on the Amazon rain forest and then use it to retrieve a map or a picture of the rain forest. In this respect, a hypertext database is like a web of interconnected data you can trace from one data item to another. Information is displayed in pages known as *web pages*. On a web page, certain keywords or graphics are highlighted that form links to other web pages or to items, such as pictures, articles, or files.

On your Linux system, you can choose from several web browsers, including Firefox, Konqueror, Epiphany, and Lynx. Firefox, Konqueror, and Epiphany are X Window System–based browsers that provide full picture, sound, and video display capabilities. Most distributions also include the Lynx browser, a line-mode browser that displays only lines of text. The K Desktop incorporates web browser capabilities into its file manager, letting a directory window operate as a web browser. GNOME-based browsers, such as Express and Mnemonic, are also designed to be easily enhanced.

Web browsers and FTP clients are commonly used to conduct secure transactions such as logging in to remote sites, ordering items, or transferring files. Such operations are currently secured by encryption methods provided by the Secure Sockets Layer (SSL). If you use a browser for secure transactions, it should be SSL enabled. Most browsers such as Mozilla and ELinks include SSL support. For FTP operations, you can use the SSH version of ftp, **sftp**, or the Kerberos 5 version. Linux distributions include SSL as part of a standard installation.

URL Addresses

An Internet resource is accessed using a Universal Resource Locator (URL), which is composed of three elements: the transfer protocol, the hostname, and the pathname. The transfer protocol and the hostname are separated by a colon and two slashes, **://**. The *pathname* always begins with a single slash:

```
transfer-protocol://host-name/path-name
```

The *transfer protocol* is usually HTTP (Hypertext Transfer Protocol), indicating a web page. Other possible values for transfer protocols are **ftp** and **file**. As their names suggest, **ftp** initiates FTP sessions, whereas **file** displays a local file on your own system, such as a text or HTML file. Table 14-1 lists the various transfer protocols.

The *hostname* is the computer on which a particular website is located. You can think of this as the address of the website. By convention, most hostnames begin with **www**. In the next example, the URL locates a web page called **guides.html** on the **tldp.org** website:

```
http://tldp.org/guides.html
```

If you do not want to access a particular web page, you can leave the file reference out, and then you automatically access the website's home page. To access a website directly, use its hostname. If no home page is specified for a website, the file **index.html** in the top directory is often used as the home page. In the next example, the user brings up the GNOME home page:

```
http://www.gnome.org/
```

The pathname specifies the directory where the resource can be found on the host system, as well as the name of the resource's file. For example, **/pub/Linux/newdat.html** references an HTML document called **newdat** located in the **/pub/Linux** directory.

The resource file's extension indicates the type of action to be taken on it. A picture has a **.gif** or **.jpeg** extension and is converted for display. A sound file has an **.au** or **.wav** extension and is played. The following URL references a **.gif** file. Instead of displaying a web page, your browser invokes a graphics viewer to display the picture. Table 14-2 provides a list of the more common file extensions.

```
http://www.train.com/engine/engine1.gif
```

Web Browsers

Most web browsers are designed to access several different kinds of information. Web browsers can access a web page on a remote website or a file on your own system. Some browsers can also access a remote news server or an FTP site. The type of information for

Protocol	Description
http	Uses Hypertext Transfer Protocol for website access.
ftp	Uses File Transfer Protocol for anonymous FTP connections.
telnet	Makes a Telnet connection.
news	Reads Usenet news; uses Network News Transfer Protocol (NNTP).

TABLE 14-1 Web Protocols

File Type	Description
.html	Web page document formatted using HTML, the Hypertext Markup Language
Graphics Files	
.gif	Graphics, using GIF compression
.jpeg	Graphics, using JPEG compression
.png	Graphics, using PNG compression (Portable Network Graphics)
Sound Files	
.au	Sun (Unix) sound file
.wav	Microsoft Windows sound file
.aiff	Macintosh sound file
Video Files	
.QT	QuickTime video file, multiplatform
.mpeg	Video file
.avi	Microsoft Windows video file

TABLE 14-2 Web File Types

a site is specified by the keyword **http** for websites, **nntp** for news servers, **ftp** for FTP sites, or **file** for files on your own system. As noted previously, several popular browsers are available for Linux. Three distinctive ones are described here: Mozilla, Konqueror, and Lynx. Mozilla is an X Window System–based web browser capable of displaying graphics, video, and sound, as well as operating as a newsreader and mailer. Konqueror is the K Desktop file manager. KDE has integrated full web-browsing capability into the Konqueror file manager, letting you seamlessly access the Web and your file system with the same application. Lynx and ELinks are command line–based browsers with no graphics capabilities, but in every other respect they are fully functional web browsers.

To search for files on FTP sites, you can use search engines provided by websites, such as Yahoo!, Google, or Lycos. These usually search for both web pages and FTP files. To find a particular web page you want on the Internet, you can use any of these search engines or perform searches from any number of web portals. Web searches have become a standard service of most websites. Searches carried out on documents within a website may use local search indexes set up and maintained by indexing programs like ht://Dig. Sites using ht://Dig use a standard web page search interface. Hypertext databases are designed to access any kind of data, whether it is text, graphics, sound, or even video. Whether you can actually access such data depends to a large extent on the type of browser you use.

The Mozilla Framework

The Mozilla project is an open source project based on the original Netscape browser code that provides a development framework for web-based applications, primarily the web browser and email client. Originally, the aim of the Mozilla project was to provide an end-user web browser called Mozilla. Its purpose has since changed to providing a development

Website	Description
mozilla.org	The Mozilla project
mozdev.org	Mozilla plug-ins and extensions
oreillynet.com/mozilla	Mozilla documentation and news
mozillazine.org	Mozilla news and articles
mozillanews.org	Mozilla news and articles
bugzilla.org	Mozilla bug reporting and tracking system

Table 14-3 Mozilla Resources

framework that anyone can use to create web applications, though the project also provides its own. Table 14-3 lists some Mozilla resources.

Currently the framework is used for Mozilla products like the Firefox web browser and the Thunderbird mail client, as well for non-Mozilla products like the Netscape, Epiphany, and Galeon web browsers. In addition, the framework is easily extensible, supporting numerous add-ons in the form of plug-ins and extensions. The Mozilla project site is **mozilla.org**, and the site commonly used for plug-in and extension development is **mozdev.org**.

The first-generation product of the Mozilla project was the Mozilla web browser, which is still available. Like the original Netscape, it included a mail client and newsreader, all in one integrated interface. The second generation products have split this integrated package into separate standalone applications, the Firefox web browser and the Thunderbird email/newsreader client. Also under development is the Camino web browser for Mac OS X and the Sunbird calendar application.

In 1998, Netscape made its source code freely available under the Netscape Public License (NPL). Mozilla is developed on an open source model much like Linux, KDE, and GNOME. Developers can submit modifications and additions over the Internet to the Mozilla website. Mozilla releases are referred to as Milestones, and Mozilla products are currently released under both the NPL license for modifications of mozilla code and the MPL license (Mozilla Public License) for new additions.

The Firefox Web Browser

Firefox is the next generation of browsers based on the Netscape core source code known as mozilla. In current releases, most distributions use Firefox as its primary browser, in place of Netscape. Firefox is a streamlined browser featuring fast web access and secure protection from invasive spyware.

Firefox is an X Window System application you can operate from any desktop, including GNOME, KDE, and XFce. Firefox is installed by default with both a menu entry in the Main menu's Internet menu and an icon on the different desktop panels. When opened, Firefox displays an area at the top of the screen for entering a URL address and a series of buttons for various web page operations like page navigation. Drop-down menus on the top menu bar provide access to such Firefox features as Tools, View, and Bookmarks.

To the right of the URL box is a search box where you can use different search engines for searching the Web, selected sites, or particular items. A pop-up menu lets you select a search engine. Currently included are Google, Yahoo, Amazon, and eBay, along with

Dictionary.com for looking up word definitions. Firefox also features button links and tabbed pages. You can drag the URL from the URL box to the button link bar to create a button with which to quickly access the site. Use this for frequently accessed sites.

For easy browsing, Firefox features tabbed panels for displaying web pages. To open an empty tabbed panel, press CTRL-T or select New Tab from the File menu. To display a page in that panel, drag its URL from the URL box or from the bookmark list to the panel. You can have several panels open at once, moving from one page to the next by clicking their tabs. You can elect to open all your link buttons as tabbed panels by right-clicking the link bar and selecting Open In Tabs.

Firefox refers to the URLs of web pages you want to keep as *bookmarks,* marking pages you want to access directly. The Bookmarks menu enables you add your favorite web pages. You can then view a list of your bookmarks and select one to view. You can also edit your list, adding new ones or removing old ones. History is a list of previous URLs you have accessed. The URL box also features a pop-up menu listing your previous history sites. Bookmarks and History can be viewed as sidebars, selectable from the View menu.

When you download a file using Firefox, the download is managed by the Download Manager. You can download several files at once. Downloads can be displayed in the Download Manager toolbar. You can cancel a download at any time, or just pause a download, resuming it later. Right-clicking a download entry will display the site it was downloaded from as well as the directory you saved it in. To remove an entry, click Remove from toolbar.

The Preferences menu (Edit | Preferences) in Firefox enables you to set several different options. Firefox also supports such advanced features as cookie, form, image, and password management. You can elect to suppress cookies from sites, automatically fill in forms, not display site images, and set up login information such as usernames and passwords for selected sites. You can set preferences for general features, privacy, web, and download management, as well as advanced features. In General preferences, you can determine your home page, page fonts, and colors, as well as connection settings such as proxy information. For Privacy you can control information saved (such as the number of history sites to remember and the download history), set policy for saving cookies, and set the size of your cache. All of these you can manually clear. Under web Features you can control pop-ups, allow software installs, and enable JavaScript. The Download Manager panel lets you configure your downloading operations, letting you specify a default download directory, whether to automatically prompt for one, and what plug-ins you may want run automatically on certain kinds of files, such as Adobe Acrobat for Adobe PDF files. The Advanced panel lets you control more complex features of browsing such as scrolling, security levels, and certificate management.

If you are on a network that connects to the Internet through a firewall, you must use the Proxies screen to enter the address of your network's firewall gateway computer. A *firewall* is a computer that operates as a controlled gateway to the Internet for your network. Several types of firewalls exist. The most restrictive kinds of firewalls use programs called *proxies,* which receive Internet requests from users and then make those requests on their behalf. There is no direct connection to the Internet.

The K Desktop File Manager: Konqueror

If you are using the K Desktop, you can use a file manager window as a web browser. The K Desktop's file manager is automatically configured to act as a web browser. It can display web pages, including graphics and links. The K Desktop's file manager supports standard web

page operations, such as moving forward and backward through accessed pages. Clicking a link accesses and displays the web page referenced. In this respect, the Web becomes seamlessly integrated into the K Desktop.

GNOME Web Browsers: Nautilus, Galeon, and Epiphany

Several other GNOME-based web browsers are also available. Epiphany, Galeon, and Kazehakase support standard web operations. Epiphany is a GNOME web browser designed to be fast with a simple interface, and it works well as a simple browser with a clean interface. It is also integrated with the desktop, featuring a download applet that will continue even after closing Epiphany. In addition, Epiphany supports tabbed panels for multiple website access. You can find out more about Epiphany at **epiphany.mozdev.org**. Galeon is a fast, light browser also based on the Mozilla browser engine (Gecko). Kazehakase emphasizes a customizable interface with download boxes and RSS bookmarks.

For GNOME, you can also download numerous support tools, such as the RSSOwl to display news feeds and the GNOME Download Manager (Gwget) for controlling web-based downloads. The Downloader for X client is useful for both FTP and web file downloads. It has numerous features, letting you control download speeds as well as download subdirectories.

Lynx and ELinks: Line-Mode Browsers

Lynx is a line-mode browser you can use without the X Window System. A web page is displayed as text only. A text page can contain links to other Internet resources but does not display any graphics, video, or sound. Except for the display limitations, Lynx is a fully functional web browser. You can use Lynx to download files or to make Telnet connections. All information on the Web is still accessible to you. Because it does not require much of the overhead that graphics-based browsers need, Lynx can operate much faster, quickly displaying web page text. To start the Lynx browser, you enter **`lynx`** on the command line and press ENTER.

Another useful text-based browser shipped with most distributions is ELinks. ELinks is a powerful screen-based browser that includes features such as frame, form, and table support. It also supports SSL secure encryption. To start ELinks, enter the **`elinks`** command in a terminal window.

Creating Your Own Website

To create your own website, you need access to a web server. Most Linux distributions automatically install the Apache web server on their Linux systems. You can also rent web page space on a remote server—a service many ISPs provide, some for free. The directory set up by your Apache web server for your website pages is usually at **/var/httpd/html**. Other servers provide you with a directory for your home page in which to place the web pages you create. You place your home page in that directory, then make other subdirectories with their own web pages to which they can link. Web pages are not difficult to create. Links from one page to another move users through your website. You can even create links to web pages or resources on other sites. Many excellent texts are available on web page creation and management.

Web pages are created using either HTML or the newer extended version, XML, the Extended Markup Language. Both are a subset of Standard Generalized Markup Language (SGML). Creating an HTML or XML document is a matter of inserting HTML or XML tags in a text file. In this respect, creating a web page is as simple as using a tag-based word processor. You use the HTML tags to format text for display as a web page. XML tags can

include more detailed information about a particular connection, such as object data or transaction characteristics. The web page itself is a text file you can create using any text editor, such as Vi. If you are familiar with tag-based word processing on Unix systems, you will find it conceptually similar to nroff. Some HTML tags indicate headings, lists, and paragraphs, as well as links to reference web resources.

Instead of manually entering HTML or XML code, you can use web page composers. A web page composer provides a graphical interface for constructing web pages. Special web page creation programs can easily help you create complex web pages without ever having to type any HTML tags explicitly. Remember, though, no matter what tool you use to create your web page, the web page itself will be an HTML document. As part of the KDE project, KDE web Dev (**kdewebdev.org**) provides several web development applications, such as the Quanta Plus web editor and the Kommander dialog builder.

NOTE *Many of the standard editors for the K Desktop and GNOME include web page construction features. Many enable you to insert links or format headings. For example, the KEdit program supports basic text-based web page components. You can add headings, links, or lines, but not graphics.*

Java for Linux

To develop Java applications, use Java tools, and run many Java products, you must install the Java 2 Software Development Kit (SDK) and the Java 2 Runtime Environment (JRE) on your system. Together, they make up the Java 2 Platform, Standard Edition (J2SE). Sun currently supports and distributes Linux versions of these products. You can download them from Sun at **java.sun.com/j2se** and install them on your system. Java packages and applications are listed in Table 14-4.

Sun, Java-like, JPackage, and Blackdown

Many Linux distributions include numerous free Java applications and support, like Jakarta, many of which were originally developed by the JPackage Project (**jpackage.org**) for use on Linux. You should use your distribution's versions of these packages, as they may have been specially modified for use on it. However, the main Java Runtime Environment and SDK are not included. Instead, you either use an included compatible set of GNU packages (Java-like) that allow you to run Java applets, or install the JRE and SDK Linux ports from either JPackage or Blackdown, or download and install the original JRE and SDK from Sun. None of these options are exclusive. The JPackage service is available for RPM based distributions like Red Hat, SUSE, and Fedora, and the Blackdown package is popular with distributions like Debian and Ubuntu.

Moat distributions like Ubuntu, Fedora, and SUSE now include a Java-like collection of support packages that enable the use of Java Runtime operations. There is no official name for this collection, though it is usually referred to as java-gci-compat, as well as Java-like. This collection provides a free and open source environment, consisting of three packages: GNU Java runtime (libgcj), the Eclipse Java compiler (ecj), and, on Fedora, a set of wrappers and links (java-gcj-compat).

Though Sun supports Linux versions of Java, more thorough and effective Linux ports of Java can be obtained from the Blackdown project at **www.blackdown.org**. Distributions like

Application	Description
Java 2 Software Development Kit (SDK)	A Java development environment with a compiler, interpreters, debugger, and more: **java.sun.com/j2se**. Part of the Java 2 platform. Download the Linux port from JPackage or Blackdown, or directly from Sun.
Java 2 Runtime Environment 1.4 (J2RE)	A Java Runtime Environment used to run Java applets. Download the Linux port from **java.com**, **blackdown.org**, **jpackage.org**.
Java 3D for Linux	Sun's 3-D Application Program Interface for 3-D Java programs.
Java-like environment	The Java-like Free and Open Environment, consisting of the GNU Java runtime (libgcj), the Eclipse Java compiler (ecj), and supporting wrappers and links (java-gcj-compat).
Java Advanced Imaging (JAI) for Linux	Java Advanced Imaging API.
Java 1.1 Development Kit (JDK) and Java 1.1 Runtime Environment (JRE) for Linux	The older Java 1.1 development environment with a compiler, interpreters, debugger, and more. Download the Linux port for your distribution's update through **blackdown.org** or **jpackage.org**.
Java System web server	A web server implemented with Java. Available at the Java website: **java.sun.com** (commercial).
GNU Java Compiler	GNU public licensed Java Compiler (GJC) to compile Java programs: **gcc.gnu.org/java**, libgjc.
Jakarta Project	Apache Software Foundation project for open source Java applications: **jakarta.apache.org**.

Table 14-4 Java Packages and Java Web Applications

Debian and Ubunto are compatible with and provide access to the Blackdown Java ports. The Blackdown project has ported the both the Java JRE and SDK, as well as previous versions of Java.

RPM based Linux distributions, like Fedora and SUSE, recommend that you download Java and Java applications from JPackage, which designs its Java packages so that they do not conflict and are compatible with the specified Linux release. With JPackage versions of Java, you can safely install and uninstall the numerous Java applications and support tools. The best way to use JPackage is to download its Yellowdog Updater Modified (Yum) repository configuration file. Be sure to select the one for the appropriate distribution (when available). Unfortunately, because of licensing restrictions, JPackage has to charge for the main JRE as well as the SDK Java application. These are included in their Java package.

Note *See **java.sun.com/products** for an extensive listing of Java applications.*

Installing the Java Runtime Environment: JRE

Many websites will run applications that require the Java Runtime Environment (JRE). Many Linux distributions do not come with the Java Runtime Environment already installed, but Java-like already supports compatible applications like Office.org and Eclipse. For Sun's Java, you will have to download and install the JRE on your Linux system. You can obtain a copy from the Sun Java site (**java.com**) . The SDK and JRE are available in the form of self-extracting compressed archives, **.bin**. These files are actually shell scripts with an embedded compressed archive. (Separate install instructions are available.) Because of filename conflicts, you should not use the Sun RPM package (**.bin.rpm**). Instead you should download the **.bin** package and extract it in the **/opt** directory. You will have to make the self-extracting bin file executable with the `chmod` command. The following command will change the JRE file, **jre-1_5_0_02-linux-i586-rpm.bin**, to an executable. You will be prompted first to accept the license agreement.

```
chmod a+x jre-1_5_0_02-linux-i586-rpm.bin
./ jre-1_5_0_02-linux-i586-rpm.bin
```

The JRE will be installed in the **/opt** directory, in this case under **/opt/jre-1_5_0_02**.

Enabling the Java Runtime Environment for Mozilla/Firefox

To allow either the Mozilla or Firefox web browser to use the JRE, you need to create a link from the Mozilla plug-in directory to the Java plug-in libraries. Be sure you have first installed the JRE. Within the **/usr/lib/mozilla/plugins** directory, you will have to create a link to the **libjavaplugin_oji.so** library in the JRE's **/plugin/i386/ns7** subdirectory, where "ns7" indicates Netscape 7.

```
# cd /usr/lib/mozilla/plugins
# ln -s /opt/jre1.5.0_06/plugin/i386/ns7/libjavaplugin_oji.so libjavaplugin_oji.so
```

NOTE *On Firefox and Mozilla, be sure Java support is enabled.*

The Java Applications

Numerous additional Java-based products and tools are currently adaptable for Linux. Tools include Java 3D, Java Media Framework (JMF), and JAI. Many of the products such as the Java web server run directly as provided by Sun. You can download several directly from the Sun Java website at **java.sun.com**. The Jakarta project (**jakarta.apache.org)**, part of the Apache Software Foundation, provides open source Java tools and applications, including libraries, server applications, and engines. Jakarta, along with other packages, is included with most distributions.

The Java 2 Software Development Kit

The Java 2 SDK provides tools for creating and debugging your own Java applets and provides support for Java applications. The kit includes demonstration applets with source code. You can obtain detailed documentation about the SDK from the Sun website at **java.sun.com**. Four major releases of the SDK are currently available—1.2, 1.3, 1.4.*x*, and 1.5 (also known as 5.0)—with

corresponding versions for the Java 2 Runtime Environment (J2RE) for 1.2, 1.3, 1.4, and 1.5 (5.0). The Java SDK adds capabilities for security, graphical user interface (GUI) support with JFC (also known as Swing), and running Java enhancements, such as Java 3D and Java Sound.

The SDK includes standard features found in the JDK features for internationalization, signed applets, the JAR file format, AWT (window toolkit) enhancements, the JavaBeans component model, networking enhancements, a math package for large numbers, database connectivity (JDBC), object serialization, and inner classes. Java applications include a Java compiler (javac), a Java debugger (jdb), and an applet viewer (appletviewer). Detailed descriptions of these features can be found in the SDK documentation, **java.sun.com/docs**.

You create a Java applet much as you would create a program using a standard programming language. You first use a text editor to create the source code, which is saved in a file with a **.java** extension. Then you can use the **javac** compiler to compile the source code file, generating a Java applet. Numerous integrated development environment (IDE) applications are available for composing Java applets and applications. Although most are commercial, some provide free shareware versions. An IDE provides a GUI for constructing Java applets. Eclipse can operate as a development platform for Java applets.

FTP Clients

With FTP clients, you can connect to a corresponding FTP site and download files from it. FTP clients are commonly used to download software from public FTP sites that operate as software repositories. Most Linux software applications can be downloaded to your Linux system from such sites, which feature anonymous logins that let any user access their files. A distribution site like **ftp.redhat.com** is an example of one such FTP site, holding an extensive set of packaged Linux applications you can download using an FTP client and then easily install on your system. Basic FTP client capabilities are incorporated into the Konqueror (KDE) and Nautilus (GNOME) file managers. You can use a file manager window to access an FTP site and drag files to local directories to download them. Effective FTP clients are also now incorporated into most web browsers, making web browsers a primary downloading tool. Firefox in particular has strong FTP download capabilities.

Though file managers and web browsers provide effective access to public (anonymous login) sites, to access private sites, you may need a standalone FTP client such as curl, wget, gFTP, or ftp. These clients let you enter usernames and passwords with which you can access a private FTP site. The standalone clients are also useful for large downloads from public FTP sites, especially those with little or no web display support. Popular Linux FTP clients are listed in Table 14-5.

Network File Transfer: FTP

With FTP clients, you can transfer extremely large files directly from one site to another. FTP can handle both text and binary files. This is one of the TCP/IP protocols, and it operates on systems connected to networks that use the TCP/IP protocols, such as the Internet. FTP performs a remote login to another account on another system that you connect to through your network. Once logged in to that other system, you can transfer files to and from it. To log in, you need to know the login name and password for the account on the remote system. For example, if you have accounts at two different sites on the Internet, you can use

FTP Client	Description
Firefox	Mozilla web and FTP browser
Konqueror	K Desktop file manager
Nautilus	GNOME file manager
gFTP	GNOME FTP client
ftp	Command line FTP client
lftp	Command line FTP client capable of multiple connections
NcFTP	Screen-based FTP client
curl	Internet transfer client (FTP and HTTP)

TABLE 14-5 Linux FTP Clients

FTP to transfer files from one to the other. However, many sites on the Internet allow public access using FTP. Such sites serve as depositories for large files that anyone can access and download. These sites are often referred to as *FTP sites,* and in many cases, their Internet addresses usually begin with the word *ftp,* such as **ftp.gnome.org** or **ftp.redhat.com**. These public sites allow anonymous FTP logins from any user. For the logins name, use the word "anonymous" and for the password, use your email address. You can then transfer files from that site to your own system.

You can perform FTP operations using an FTP client program; for Linux systems, you can choose from several FTP clients. Many now operate using GUIs such as GNOME. Some, such as Firefox, have limited capabilities, whereas others, such as NcFTP, include an extensive set of enhancements. The original FTP client, ftp, is just as effective, though not as easy to use. It operates using a simple command line interface and requires no GUI or cursor support, as other clients do.

The Internet has a great many sites open to public access that contain files anyone can obtain using file transfer programs. Unless you already know where a file is located, however, finding it can be difficult. To search for files on FTP sites, you can use search engines provided by websites, such as Yahoo!, Google, or Lycos. For Linux software, you can check sites such as **freshmeat.net, sourceforge.net**, **rpmfind.net**, **freshrpms.net**, **apps.kde.com**, and **gnome.org**. These sites usually search for both web pages and FTP files.

Web Browser–Based FTP: Firefox

You can access an FTP site and download files from it with any web browser. A web browser is effective for checking out an FTP site to see what files are listed there. When you access an FTP site with a web browser, the entire list of files in a directory is listed as a web page. You can move to a subdirectory by clicking its entry. With Firefox, you can easily browse through an FTP site to download files: just click the download link. This will start the transfer operation, opening a box for selecting your local directory and the name for the file. The default name is the same as on the remote system. You can manage your downloads with the Download Manager, which will let you cancel a download operation in progress or remove other downloads requested. The manager will show the time remaining,

the speed, and the amount transferred for the current download. Browsers are useful for locating individual files, though not for downloading a large set of files, as is usually required for a system update.

The K Desktop File Manager: Konqueror

On the K Desktop, the desktop file manager (Konqueror) has a built-in FTP capability. The FTP operation has been seamlessly integrated into standard desktop file operations. Downloading files from an FTP site is as simple as copying files by dragging them from one directory window to another, but one of the directories happens to be located on a remote FTP site. On the K Desktop, you can use a file manager window to access a remote FTP site. Files in the remote directory are listed just as your local files are. To download files from an FTP site, you open a window to access that site, entering the URL for the FTP site in the window's location box. Open the directory you want, and then open another window for the local directory to which you want the remote files copied. In the window showing the FTP files, select the ones you want to download. Then simply click and drag those files to the window for the local directory. A pop-up menu appears with choices for Copy, Link, or Move. Select Copy. The selected files are then downloaded. Another window then opens, showing the download progress and displaying the name of each file in turn, along with a bar indicating the percentage downloaded so far.

GNOME Desktop FTP: Nautilus

The easiest way to download files is to use the built-in FTP capabilities of the GNOME file manager, Nautilus. The FTP operation has been seamlessly integrated into standard desktop file operations. Downloading files from an FTP site is as simple as dragging files from one directory window to another, where one of the directories happens to be located on a remote FTP site. Use the GNOME file manager to access a remote FTP site, listing files in the remote directory, just as local files are. Just enter the FTP URL following the prefix **ftp://** and press ENTER. The top directory of the remote FTP site will be displayed. Simply use the file manager to progress through the remote FTP site's directory tree until you find the file you want. Then open another window for the local directory to which you want the remote files copied. In the window showing the FTP files, select those you want to download. Then CTRL-click and drag those files to the window for the local directory. CTRL-clicking performs a copy operation, not a move. As files are downloaded, a dialog window appears, showing the progress.

gFTP

The gFTP program is a simpler GNOME FTP client designed to let you make standard FTP file transfers. The gFTP window consists of several panes. The top-left pane lists files in your local directory, and the top-right pane lists your remote directory. Subdirectories have folder icons preceding their names. The parent directory can be referenced by the double period entry (**..**) with an up arrow at the top of each list. Double-click a directory entry to access it. The pathnames for all directories are displayed in boxes above each pane. You can enter a new pathname for a different directory to change to it, if you want.

Two buttons between the panes are used for transferring files. The left arrow button (<-) downloads selected files in the remote directory, and the right arrow button (->) uploads files from the local directory. To download a file, click it in the right-side pane and then click

the left arrow button. When the file is downloaded, its name appears in the left pane, your local directory. Menus across the top of the window can be used to manage your transfers. A connection manager enables you to enter login information about a specific site. You can specify whether to perform an anonymous login or to provide a username and password. Click Connect to connect to that site. A drop-down menu for sites enables you to choose the site you want. Interrupted downloads can be restarted easily.

wget

The wget tool lets you easily access web and FTP sites for particular directories and files. Directories can be recursively downloaded, letting you copy an entire website. **wget** takes as its option the URL for the file or directory you want. Helpful options include **-q** for quiet, **-r** for recursive (directories), **-b** to download in the background, and **-c** to continue downloading an interrupted file. One of the drawbacks is that your URL reference can be very complex. You have to know the URL already; you cannot interactively locate an item as you would with an FTP client. The following would download the Fedora DVD in the background:

```
wget -b  ftp://download.fedora.redhat.com/pub/fedora/linux/core/7/i386/iso/
FC-7-i386-DVD.iso
```

TIP *With the GNOME wget tool, you can run wget downloads using a GUI.*

curl

The curl Internet client operates much like wget but with much more flexibility. You can specify multiple URLs on **curl**'s command line, and you can use braces to specify multiple matching URLs, such as different websites with the same domain name. You can list the different website host names within braces followed by their domain name (or vice versa). You can also use brackets to specify a range of multiple items. This can be very useful for downloading archived files that have the same root name with varying extensions, such as different issues of the same magazine. curl can download using any protocol and will try to intelligently guess the protocol to use if none is given. Check the **curl** Man page for more information.

ftp

The name ftp designates the original FTP client used on Unix and Linux systems. The ftp client uses a command line interface, and it has an extensive set of commands and options you can use to manage your FTP transfers. You start the ftp client by entering the command **ftp** at a shell prompt. If you have a specific site you want to connect to, you can include the name of that site on the command line after the **ftp** keyword. Otherwise, you need to connect to the remote system with the ftp command **open**. You are then prompted for the name of the remote system with the prompt "(to)". When you enter the remote system name, ftp connects you to the system and then prompts you for a login name. The prompt for the login name consists of the word "Name" and, in parentheses, the system name and your local login name. Sometimes the login name on the remote system is the same as the login name on your own system. If the names are the same, press ENTER at the prompt. If they are

different, enter the remote system's login name. After entering the login name, you are prompted for the password. In the next example, the user connects to the remote system **garnet** and logs in to the **robert** account:

```
$ ftp
ftp> open
(to) garnet
Connected to garnet.berkeley.edu.
220 garnet.berkeley.edu FTP server ready.
Name (garnet.berkeley.edu:root): robert
password required
Password:
user robert logged in
ftp>
```

Once logged in, you can execute Linux commands on either the remote system or your local system. You execute a command on your local system in ftp by preceding the command with an exclamation point. Any Linux commands without an exclamation point are executed on the remote system. One exception exists to this rule: whereas you can change directories on the remote system with the **cd** command, to change directories on your local system, you need to use a special ftp command called **lcd** (local **cd**). In the next example, the first command lists files in the remote system, while the second command lists files in the local system:

```
ftp> ls
ftp> !ls
```

The ftp program provides a basic set of commands for managing files and directories on your remote site, provided you have the permission to do so (see Table 14-6). You can use **mkdir** to create a remote directory and **rmdir** to remove one. Use the **delete** command to erase a remote file. With the **rename** command, you can change the names of files. You close your connection to a system with the **close** command. You can then open another connection if you want. To end the ftp session, use the **quit** or **bye** command.

```
ftp> close
ftp> bye
Good-bye
$
```

To transfer files to and from the remote system, use the **get** and **put** commands. The **get** command receives files from the remote system to your local system, and the **put** command sends files from your local system to the remote system. In a sense, your local system gets files *from* the remote and puts files *to* the remote. In the next example, the file **weather** is sent from the local system to the remote system using the **put** command:

```
ftp> put weather
PORT command successful.
ASCII data connection
ASCII Transfer complete.
ftp>
```

Command	Effect
ftp	Invokes the ftp program.
open *site-address*	Opens connection to another system.
close	Closes connection to a system.
quit or **bye**	Ends ftp session.
ls	Lists the contents of a directory.
dir	Lists the contents of a directory in long form.
get *filename*	Sends file from remote system to local system.
put *filename*	Sends file from local system to remote system.
mget *regular-expression*	Enables you to download several files at once from a remote system. You can use special characters to specify the files; you are prompted to transfer each file in turn.
mput *regular-expression*	Enables you to send several files at once to a remote system. You can use special characters to specify the files; you are prompted for each file to be transferred.
runique	Toggles storing of files with unique filenames. If a file already exists with the same filename on the local system, a new filename is generated.
reget *filename*	Resumes transfer of an interrupted file from where you left off.
binary	Transfers files in binary mode.
ascii	Transfers files in ASCII mode.
cd *directory*	Changes directories on the remote system.
lcd *directory*	Changes directories on the local system.
help or **?**	Lists ftp commands.
mkdir *directory*	Creates a directory on the remote system.
rmdir	Deletes a remote directory.
delete *filename*	Deletes a file on the remote system.
mdelete *file-list*	Deletes several remote files at once.
rename	Renames a file on a remote system.
hash	Displays progressive hash signs during download.
status	Displays current status of ftp.

TABLE 14-6 The ftp Client Commands

If a download is ever interrupted, you can resume the download with **reget**. This is helpful for an extremely large file. The download resumes from where it left off, so the whole file needn't be downloaded again. Also, be sure to download binary files in binary mode. For most FTP sites, the binary mode is the default, but some sites might have ASCII (text) as

the default. The command **ascii** sets the character mode, and the command **binary** sets the binary mode. Most software packages available at Internet sites are archived and compressed files, which are binary files. In the next example, the transfer mode is set to binary, and the archived software package **mydata.tar.gz** is sent from the remote system to your local system using the **get** command:

```
ftp> binary
ftp> get mydata.tar.gz
PORT command successful.
Binary data connection
Binary Transfer complete.
ftp>
```

You may often want to send several files, specifying their names with wildcard characters. The **put** and **get** commands, however, operate only on a single file and do not work with special characters. To transfer several files at a time, you have to use two other commands, **mput** and **mget**. When you use **mput** or **mget**, you are prompted for a file list. You can then either enter the list of files or a file-list specification using special characters. For example, ***.c** specifies all the files with a **.c** extension, and ***** specifies all files in the current directory. In the case of **mget**, files are sent one by one from the remote system to your local system. Each time, you are prompted with the name of the file being sent. You can type **y** to send the file or **n** to cancel the transmission. You are then prompted for the next file. The **mput** command works in the same way, but it sends files from your local system to the remote system. In the next example, all files with a **.c** extension are sent to your local system using **mget**:

```
ftp> mget
(remote-files) *.c
mget calc.c? y
PORT command successful
ASCII data connection
ASCII transfer complete
mget main.c? y
PORT command successful
ASCII data connection
ASCII transfer complete
ftp>
```

Answering the prompt for each file can be a tedious prospect if you plan to download a large number of files, such as those for a system update. In this case, you can turn off the prompt with the **prompt** command, which toggles the interactive mode on and off. The **mget** operation then downloads all files it matches, one after the other.

```
ftp> prompt
Interactive mode off.
ftp> mget
(remote-files) *.c
 PORT command successful
ASCII data connection
ASCII transfer complete
```

```
PORT command successful
ASCII data connection
ASCII transfer complete
ftp>
```

NOTE *To access a public FTP site, you have to perform an anonymous login. Instead of a login name, you enter the keyword* ***anonymous*** *(or* ***ftp****). Then, for the password, you enter your email address. Once the ftp prompt is displayed, you are ready to transfer files. You may need to change to the appropriate directory first or set the transfer mode to binary.*

Automatic Login and Macros: .netrc

The ftp client has an automatic login capability and support for macros. Both are entered in a user's ftp configuration file called **.netrc**. Each time you connect to a site, the **.netrc** file is checked for connection information, such as a login name and password. In this way, you needn't enter a login name and password each time you connect to a site. This feature is particularly useful for anonymous logins. Instead of having to enter the username **anonymous** and your email address as your password, this information can be automatically read from the **.netrc** file. You can even make anonymous login information your default so that, unless otherwise specified, an anonymous login is attempted for any FTP site to which you try to connect. If you have sites you must log in to, you can specify them in the **.netrc** file and, when you connect, either automatically log in with your username and password for that site or be prompted for them.

Entries in the **.netrc** file have the following syntax. An entry for a site begins with the term "machine," followed by the network or Internet address, and then the login and password information.

```
machine system-address login remote-login-name password password
```

The following example shows an entry for logging in to the **dylan** account on the **turtle.trek.com** system:

```
machine turtle.trek.com login dylan password legogolf
```

For a site you would anonymously log in to, you enter the word "anonymous" for the login name and your email address for the password.

```
machine ftp.redhat.com login anonymous password dylan@turtle.trek.com
```

In most cases, you'll use ftp to access anonymous FTP sites. Instead of trying to make an entry for each one, you can make a default entry for anonymous FTP login. When you connect to a site, ftp looks for a machine entry for it in the **.netrc** file. If none exists, ftp looks for a default entry and uses that. A default entry begins with the word "default" with no network address. To make anonymous logins your default, enter **anonymous** and your email address as your login and password.

```
default login anonymous password dylan@turtle.trek.com
```

A sample **.netrc** file with a machine definition and a default entry is shown here:

.netrc

```
machine golf.mygames.com login dylan password legogolf
default login anonymous password dylan@turtle.trek.com
```

You can also define macros in your **.netrc** file. With a macro, you can execute several ftp operations at once using only the macro name. Macros remain in effect during a connection. When you close a connection, the macros are undefined. Although a macro can be defined on your ftp command line, defining them in **.netrc** entries makes more sense. This way, you needn't redefine them again; they are read automatically from the **.netrc** file and defined for you. You can place macro definitions within a particular machine entry in the **.netrc** file or in the default entry. Macros defined in machine entries are defined only when you connect to that site. Macros in the default entry are defined whenever you make a connection to any site.

The syntax for a macro definition follows. It begins with the keyword **macdef**, followed by the macro name you want to give it, and ends with an empty line. An ftp macro can take arguments, referenced within the macro with **$n**, where **$1** references the first argument, and **$2** the second, and so on. If you need to use a **$** character in a macro, you have to quote it using the backslash, **\$**.

```
macdef macro-name
ftp commands
empty-line
```

lftp

The lftp program is an enhanced FTP client with advanced features such as the capability to download mirror sites and run several FTP operations in the background at the same time. lftp uses a command set similar to that for the ftp client: you use **get** and **mget** commands to download files, with the **-o** option to specify local locations for them. Use **lcd** and **cd** to change local and remote directories.

To manage background commands, you use many of the same commands as for the shell. The **&** placed at the end of a command puts it into the background, and CTRL-Z puts an already-running job in the background. Commands can be grouped with parentheses and placed together into the background. Use the **jobs** command to list your background jobs and the **wait** or **fg** command to move jobs from the background to the foreground. When you exit lftp, the program will continue to run any background jobs. In effect, lftp becomes a background job itself.

When you connect to a site, you can queue commands with the **queue** command, setting up a list of FTP operations to perform. This feature allows you to queue several download operations to a site. The queue can be reordered and entries deleted if you wish. You can also connect to several sites and set up a queue for each one. The **mirror** command lets you maintain a local version of a mirror site. You can download an entire site or just update newer files, as well as remove files no longer present on the mirror.

You can tailor lftp with options set in the **.lftprc** file. Systemwide settings are placed in the **/etc/lftp.conf** file. Here, you can set features like the prompt to use and your anonymous password. The **.lftp** directory holds support files for command history, logs, bookmarks, and startup commands. The lftp program also supports the **.netrc** file, checking it for login information.

NcFTP

The NcFTP program has a screen-based interface that can be run from any shell command line. It does not use a desktop interface. To start up NcFTP, you enter the **ncftp** command on the command line. If you are working in a window manager, such as KDE, GNOME, or XFce, open a shell terminal window and enter the command at its prompt. The main NcFTP screen consists of an input line at the bottom of the screen with a status line above it. The remainder of the screen displays commands and responses from remote systems. For example, when you download files, a message specifying the files to be downloaded is displayed in the status line. NcFTP lets you set preferences for different features, such as anonymous login, progress meters, or a download directory. Enter the **pref** command to open the preferences screen. From there, you can select and modify the listed preferences.

To connect to an FTP site, enter the **open** command on the input line, followed by the site's address. The address can be either an IP address or a domain name, such as **ftp.gnome.org**. If you don't supply an address, a list of your bookmarked sites is displayed, and you can choose one from there. By default, NcFTP attempts an anonymous login, using the term "anonymous" as your username and your email address as the password. When you successfully connect, the status bar displays the remote site's name on the left and the remote directory name.

If you want to log in to a specific account on a remote site, have yourself prompted for the username and password by using the **-u** option with the **open** command. The **open** command remembers the last kind of login you performed for a specific site and repeats it. If you want to change back to an anonymous login from a user login, use the **-a** option with the **open** command.

Once connected, you enter commands on the input line to perform FTP operations such as displaying file lists, changing directories, or downloading files. With the **ls** command, you can list the contents of the current remote directory. Use the **cd** command to change to another remote directory. The **dir** command displays a detailed listing of files. With the **page** command, you view the contents of a remote file, a screen at a time. To download files, use the **get** command, and to upload files, use the **put** command. During a download, a progress meter above the status bar displays how much of the file has been downloaded so far. The **get** command has several features described in more detail in the following section. When you finish, you can disconnect from the site with the **close** command. You can then use **open** to connect to another site, or quit the NcFTP program with the **quit** command. The **help** command lists all NcFTP commands. You can use the **help** command followed by the name of a command to display specific information on it.

The NcFTP **get** command differs significantly from the original FTP client's **get** command. Whereas ftp uses two commands, **get** and **mget**, to perform download operations, NcFTP uses only the **get** command. However, the NcFTP **get** command combines the capabilities of both **mget** and **get** and also adds several new features. By default, the NcFTP **get** command performs wildcard matching for filenames. If you enter only part of a filename, the **get** command tries to download all files beginning with that name. You can turn off wildcard matching with the **-G** option, in which case you must enter the full names of the files you want.

This page intentionally left blank

15 CHAPTER

Network Tools

You can use a variety of network tools to perform tasks such as obtaining information about other systems on your network, accessing other systems, and communicating directly with other users. Network information can be obtained using utilities such as **ping**, **finger**, **traceroute**, and **host**. Talk, ICQ, and IRC clients enable you to communicate directly with other users on your network. Telnet performs a remote login to an account you may have on another system connected on your network. Some tools have a corresponding K Desktop or GNOME version. In addition, your network may make use of network remote access commands. These are useful for smaller networks and enable you to access remote systems directly to copy files or execute commands.

Network Information: ping, finger, traceroute, and host

You can use the **ping**, **finger**, **traceroute**, and **host** commands to find out status information about systems and users on your network. The **ping** command is used to check if a remote system is up and running. You use **finger** to find out information about other users on your network, seeing if they are logged in or if they have received mail; **host** displays address information about a system on your network, giving you a system's IP and domain name addresses; and **traceroute** can be used to track the sequence of computer networks and systems your message passed through on its way to you. Table 15-1 lists various network information tools.

GNOME Network Tools: gnome-nettool

For the GNOME desktop, the **gnome-nettool** utility provides a GNOME interface for entering the **ping** and **traceroute** commands, among other features, including Finger, Whois, and Lookup for querying users and hosts on the network. Whois will provide domain name information about a particular domain, and Lookup will provide both domain name and IP addresses. You can access **gnome-nettool** with the Network Tools entry in the System Tools menu. It also includes network status tools such as **netstat** and **portscan**. The first panel, Devices, describes your connected network devices, including configuration and transmission information about each device, such as the hardware address and bytes transmitted. Both IPv4 and IPv6 host IP addresses will be listed.

Network Information Tool	Description
ping	Detects whether a system is connected to the network.
finger	Obtains information about users on the network.
who	Checks what users are currently online.
whois	Obtains domain information.
host	Obtains network address information about a remote host.
traceroute	Tracks the sequence of computer networks and hosts your message passes through.
wireshark	Protocol analyzer that examines network traffic.
gnome-nettool	GNOME interface for various network tools including ping, finger, and traceroute.
mtr and xmtr	My traceroute combines both ping and traceroute operations (Traceroute on System Tools menu).

TABLE 15-1 Network Information Tools

ping

The **ping** command detects whether a system is up and running. **ping** takes as its argument the name of the system you want to check. If the system you want to check is down, **ping** issues a timeout message indicating a connection could not be made. The next example checks to see if **redhat.com** is up and connected to the network:

```
# ping www.redhat.com
PING www.redhat.com (209.132.177.50) 56(84) bytes of data.
64 bytes from www.redhat.com (209.132.177.50): icmp_seq=1 ttl=118 time=36.7 ms
64 bytes from www.redhat.com (209.132.177.50): icmp_seq=2 ttl=118 time=36.9 ms
64 bytes from www.redhat.com (209.132.177.50): icmp_seq=3 ttl=118 time=37.4 ms

--- www.redhat.com ping statistics ---
4 packets transmitted, 3 received, 25% packet loss, time 3000ms
rtt min/avg/max/mdev = 36.752/37.046/37.476/0.348 ms
```

You can also use **ping** with an IP address instead of a domain name. With an IP address, **ping** can try to detect the remote system directly without having to go through a domain name server to translate the domain name to an IP address. This can be helpful for situations where your network's domain name server may be temporarily down and you want to check if a particular remote host on your network is connected. In the next example, the user checks the Red Hat site using its IP address:

```
# ping 209.132.177.50
PING 209.132.177.50 (209.132.177.50) 56(84) bytes of data.
64 bytes from 209.132.177.50: icmp_seq=1 ttl=118 time=37.4 ms
64 bytes from 209.132.177.50: icmp_seq=2 ttl=118 time=37.0 ms
64 bytes from 209.132.177.50: icmp_seq=3 ttl=118 time=36.3 ms
```

```
--- 209.132.177.50 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2001ms
rtt min/avg/max/mdev = 36.385/36.969/37.436/0.436 ms
```

NOTE *A* **ping** *operation can fail if* **ping** *access is denied by a network's firewall.*

finger and who

You can use the **finger** command to obtain information about other users on your network and the **who** command to see what users are currently online on your system. The **who** and **w** commands list all users currently connected, along with when, how long, and where they logged in. The **w** command provides more detailed information and has several options for specifying the level of detail. The **who** command is meant to operate on a local system or network; **finger** can operate on large networks, including the Internet, though most systems block it for security reasons.

NOTE *Wireshark is a protocol analyzer that can capture network packets and display detailed information about them. You can detect what kind of information is being transmitted on your network as well as its source and destination. Wireshark is used primarily for network and server administration.*

host

With the **host** command, you can find network address information about a remote system connected to your network. This information usually consists of a system's IP address, domain name address, domain name nicknames, and mail server. This information is obtained from your network's domain name server. For the Internet, this includes all systems you can connect to over the Internet.

The **host** command is an effective way to determine a remote site's IP address or URL. If you have only the IP address of a site, you can use **host** to find out its domain name. For network administration, an IP address can be helpful for making your own domain name entries in your **/etc/host** file. That way, you needn't rely on a remote domain name server (DNS) for locating a site.

```
# host gnomefiles.org
gnomefiles.org has address 67.18.254.188
gnomefiles.org mail is handled by 10 mx.zayda.net.

# host 67.18.254.188
188.254.18.67.in-addr.arpa domain name pointer gnomefiles.org.
```

traceroute

Internet connections are made through various routes, traveling through a series of interconnected gateway hosts. The path from one system to another can take different routes, some of which may be faster than others. For a slow connection, you can use **traceroute** to check the route through which you are connected to a host, monitoring the speed and the number of intervening gateway connections a route takes. The **traceroute** command takes as its argument the hostname or IP addresses for the system whose route you want to check. Options are available for specifying parameters like the type of service (**-t**) or the source

host (**-s**). The **traceroute** command will return a list of hosts the route traverses, along with the times for three probes sent to each gateway. Times greater than five seconds are displayed with an asterisk (*).

```
traceroute rabbit.mytrek.com
```

You can also use the mtr or xmtr tools to perform both ping and traces (Traceroute on the System Tools menu).

Network Talk and Messenger Clients: VoIP, ICQ, IRC, AIM, and Talk

You may, at times, want to communicate directly with other users on your network. You can do so with Voice over Internet Protocol (VoIP), Talk, ICQ, instant messenger (IM), and IRC utilities, provided the other user is also logged in to a connected system at the same time (see Table 15-2). With VoIP applications, you can speak over Internet connections as if you're on a telephone. The Talk utility operates like a two-way text messaging tool, enabling you to have a direct two-way conversation with another user. Talk is designed for users on the same system or connected on a local network. ICQ (I Seek You) is an Internet tool that notifies you when other users are online and enables you to communicate with them. ICQ works much like an IM. With an Internet Relay Chat utility (IRC), you can connect to a remote server where other users are also connected and talk with them. IM clients operate much the same way, allowing users on the same IM system to communicate anywhere across the Internet. Currently the major IM systems are AOL Instant Messenger (AIM), Microsoft Network (MSN), Yahoo, ICQ, and Jabber. Unlike the others, Jabber is an open source IM service (**jabber.org**).

Ekiga

Ekiga is GNOME's new VoIP application providing Internet IP telephone and video conferencing support. It was formerly called GnomeMeeting, and its website is still at **gnomemeeting.org**. Ekiga supports both the H.323 and SIP (Session Initiation Protocol) protocols. It is compatible with Microsoft's NetMeeting. H.323 is a comprehensive protocol that includes the digital broadcasting protocols like DVB and H.261 for video streaming, as well as the supporting protocols like the H.450 series for managing calls.

Client	Description
Ekiga	VoIP application
GnomeICU	GNOME ICQ client
X-Chat	IRC client
Konversation	KDE IRC client
Gabber	Jabber client
Gaim	GNOME AIM client
psi	Jabber client using QT (KDE)
nalm	Command line cursor-based IRC, ICQ, and AIM client

Table 15-2 Talk and Messenger Clients

To use Ekiga you will need a SIP address. You can obtain a free one from **ekiga.net**, but you will first have to subscribe to the service. When you start Ekiga, you will be prompted to configure your connection. Here you can provide information like contact information, your connection method, sound driver, and video device. Use the address book to connect to another Ekiga user. A white pages directory lets you search for people who are also using Ekiga.

ICQ and IRC

The ICQ protocol enables you to communicate directly with other users online, like an IM utility. Using an ICQ client, you can send users messages, chat with them, or send files. You can set up a contact list of users you may want to contact when they are online. You are then notified in real time when they connect, and you can communicate with them if you wish. Several modes of communication are supported. These include chat, message, email, file transfer, and games. To use ICQ, you register with an ICQ server that provides you with an ICQ number, also known as a Universal Internet Number (UIN). You can find out more about the ICQ protocol at **icq.com**.

IRC operates like a chat room, where you can enter channels and talk to other users already there. First, you select an IRC server to connect to. Various servers are available for different locales and topics. Once connected to a server, you can choose from a list of channels to enter. The interface works much like a chat room. When you connect to the server, you can choose a nickname by which you will be known. Several IRC clients are available for use on Linux systems. Most operate on either the X Window System, KDE, or GNOME platforms.

Several GNOME- and KDE-based ICQ and IRC clients are available for your use. Check the GNOME software listings at **gnomefiles.org** for new versions and recent updates. For KDE-based ICQ clients, check **kde-apps.org** (Network | Chat).

Instant Messengers

AOL Instant Messenger (AIM) is a free service provided by AOL for anyone who registers for it, as well as for those who are already members of AOL. With AIM, you can send messages to members instantly, play games with them, and receive stock alerts. You can even share images, sounds, and photographs. AOL already provides clients for Windows and Macintosh. A new version called AIM Express is designed to run on any web browser and will run on systems with JDK 1.1 or greater. You can find out more about AIM at **aim.com**.

Several GNOME instant messaging clients are designed to work with all instant messaging systems, including AIM, Yahoo, MSN, and ICQ. Gaim has plug-ins that let you connect to ICQ, Yahoo, MSN, IRC, Jabber, and Zephyr. Gabber, a Jabber client, is an open source instant messaging system that allows communication with all other systems, including AIM, Yahoo, MSN, and ICQ.

NOTE *Talk is the original Unix talk utility designed to set up an interactive two-way communication between you and another user using a command line interface. It works much like instant messenger. Because of security concerns, you should use Talk only on a locally secure system. A K Desktop version of Talk called KTalk displays user screens as panes in a K Desktop window. GNU Talk is a GNOME version of Talk that supports multiple clients, file transfers, encryption, shared applications, auto-answer, and call forwarding.*

Telnet

You use the **telnet** command to log in remotely to another system on your network. The system can be on your local area network or available through an Internet connection. Telnet operates as if you were logging in to another system from a remote terminal. You will be asked for a login name and, in some cases, a password. In effect, you are logging in to another account on another system. In fact, if you have an account on another system, you could use Telnet to log in to it.

CAUTION *The original version of Telnet is noted for being very insecure. For secure connections over a network or the Internet, you should use the Secure Shell (SSH) or Kerberos versions of Telnet. They operate in the same way as the original but use authentication and encryption to secure the Telnet connection. Even so, it is advisable never to use Telnet to log in to your root account.*

You invoke the Telnet utility with the keyword **telnet**. If you know the name of the site you want to connect with, you can enter **telnet** and the name of the site on the Linux command line. As an alternative, you can use the K Desktop KTelnet utility. This provides a GUI interface for connecting and logging in to remote systems.

```
$ telnet garnet.berkeley.edu
Connected to garnet
login:
```

The Telnet program also has a command mode with a series of commands you can use to configure your connection. You can enter the **telnet** command mode either by invoking Telnet with the keyword **telnet** or by pressing CTRL-] during a session. The Telnet **help** command lists all the Telnet commands you can use. A comprehensive list is available on the Man pages (**man telnet**). In the next example, the user first invokes the Telnet utility. A prompt is displayed next, indicating the command mode, **telnet>**. The Telnet command **open** then connects to another system.

```
$ telnet
telnet> open garnet.berkeley.edu
Connected to garnet.berkeley.edu
login:
```

Once connected, you follow the login procedure for that system. If you are logging in to a regular system, you must provide a login name and password. Once logged in, you are provided with the operating system prompt; in the case of Linux or Unix, this will be either **$** or **%**. You are then directly connected to an account on that system and can issue any commands you want. When you finish your work, you log out. This breaks the connection and returns you to the Telnet prompt on your own system. You can then quit Telnet with the **quit** command.

```
telnet> quit
```

When using Telnet to connect to a site that provides public access, you needn't provide a login name or password. Access is usually controlled by a series of menus that restrict what you can do on that system. If you are logging in to a specific account on another system, you can use the **-l** option to specify the login name of that account.

RSH, Kerberos, and SSH Remote Access Commands

The remote access commands were designed for smaller networks, such as intranets. They enable you to log in remotely to another account on another system and to copy files from one system to another. You can also obtain information about another system, such as who is logged on currently (see Table 15-3). Many of the remote commands have comparable network communication utilities used for the Internet. For example, **rlogin**, which remotely logs in to a system, is similar to **telnet**. The **rcp** command, which remotely copies files, performs much the same function as **ftp**.

Because of security risks with the original versions of the remote operations **rcp**, **rlogin**, and **rsh** (RSH package), secure implementations are now installed with most Linux distributions. Secure versions of these commands are provided by Kerberos and the SSH. The Kerberos versions are configured as the default. Whenever you enter an **rcp** or **rsh** command, you are actually invoking the Kerberos version of the command. Kerberos provides versions for Telnet, **rlogin**, **rcp**, **rsh**, and **ftp**, which provide authentication and encryption. These versions operate using the same commands and options as the originals, making their use transparent to the user. On some distributions, like Fedora, when Kerberos is installed on your system, the user's **PATH** variable is configured to access the Kerberos versions of the remote commands, located at **/usr/kerberos/bin**, instead of **/usr/bin**, making the Kerberos versions the effective defaults.

The SSH versions use slightly different names, using an initial *s* in the commands, such as **ssh**, **slogin**, or **scp**. SSH commands are encrypted, providing a very high level of security.

Remote Command	Effect
rwho	Displays all users logged in to systems in your network.
ruptime	Displays information about each system on your network.
rlogin *system-name*	Allows you to log in remotely to an account on another system. Kerberos version used by default. The **-l** option allows you to specify the login name of the account.
slogin *system-name*	Secure login to an account on another system.
rcp *sys-name:file1 sys-name:file2*	Allows you to copy a file from an account on one system to an account on another system. With the **-p** option, preserves the modification times and modes of source files. Kerberos version used by default.
scp *sys-name:file1 sys-name:file2*	Securely copies a file from an account on one system to an account on another system.
rsh *sys-name Linux-command*	Allows you to remotely execute a command on another system. The **-l** option allows you to specify the login name; **-n** redirects input from the null special device, **/dev/null**. Kerberos version used by default.
ssh *sys-name Linux-command*	Securely and remotely executes a command on another system.

Table 15-3 Remote Access Commands

Even the original remote commands now include Kerberos support, enabling them to use more secure access configurations like those provided by **.k5login**. Still, these commands can allow easy unencrypted remote access to a Linux system. They should be used only within a local secure network.

Remote Access Information

You can use several commands to obtain information about different systems on your network. You can find out who is logged in, get information about a user on another system, or find out if a system is up and running. For example, the **rwho** command functions in the same way as the **who** command. It displays all the users currently logged in to each system in your network.

```
$ rwho
violet robert:tty1 Sept 10 10:34
garnet chris:tty2 Sept 10 09:22
```

The **ruptime** command displays information about each system on your network. The information shows how each system has been performing: **ruptime** shows whether a system is up or down, how long it has been up or down, the number of users on the system, and the average load on the system for the last five, ten, and fifteen minutes.

```
$ ruptime
violet up 11+04:10, 8 users, load 1.20 1.10 1.00
garnet up 11+04:10, 20 users, load 1.50 1.40 1.30
```

Remote Access Permission: .k5login

The remote commands on many distributions are Kerberos enabled, allowing you to use Kerberos authentication to control access. For ease of use, you can use the **.k5login** file to control access to your account by users using remote commands (**.rhosts** is not used). Users create this file on their own accounts using a standard editor. They must be located in the user's home directory.

The **.k5login** file is a simple way to allow other people access to your account without giving out your password. To deny access to a user, simply delete the system's name and the user's login name from your **.k5login** file. If a user's login name and system are in an **.k5login** file, that user can directly access that account without knowing the password (in place of using **.k5login**, you could use a password). The **.k5login** file will contain Kerberos names for users, including usernames and realms. Such a user will undergo Kerberos authentication to gain access. A **.k5login** file is required for other remote commands, such as remotely copying files or remotely executing Linux commands.

The type of access **.k5login** provides enables you to use remote commands to access accounts directly that you might have on other systems. You do not have to log in to them first. In effect, you can treat your accounts on other systems as extensions of the one you are currently logged in to. Using the **rcp** command, you can copy any files from one directory to another no matter what account they are on. With the **rsh** command, you can execute any Linux command you want on any of your other accounts.

rlogin, slogin, rcp, scp, rsh, and ssh

You may have accounts on different systems in your network, or you may be permitted to access someone else's account on another system. You can access an account on another system by first logging in to your own and then remotely logging in across your network to the account on the other system. You can perform such a remote login using the **rlogin** command, which takes as its argument a system name. The command connects you to the other system and begins login procedures. Bear in mind that if you are using an SSH-enabled network connection, you could use **slogin** instead of **rlogin**. Either **slogin** or Kerberos **rlogin** will provide secure encrypted login access.

You can use the **rcp** command to copy files to and from remote and local systems. For SSH-enabled network connections, you would use **scp** instead of **rcp**. The **rcp** and **scp** commands are file transfer tools that operate like the **cp** command but across a network connection to a remote system. The **rcp** command begins with the keyword **rcp** and has as its arguments the names of the source file and the copy file. To specify the file on the remote system, you need to place the remote system name before the filename, separated from it by a colon. When you are copying a file on the remote system to your own, the source file is a remote file and requires the remote system's name. The copy file is a file on your own system and does not require a system name:

```
$ rcp remote-system-name:source-file copy-file
```

In the next example, the user copies the file **wednesday** from the remote system **violet** to her own system and renames the file **today**:

```
$ rcp violet:wednesday today
```

You can also use **scp** or **rcp** to copy whole directories to or from a remote system. The **scp** command with the **-r** option copies a directory and all its subdirectories from one system to another. Like the **cp** command, these commands require source and destination directories. The directory on the remote system requires that the system name and colon be placed before the directory name. When you copy a directory from your own system to a remote system, the copy directory is on the remote system and requires the remote system's name. In the next example, the user uses the **scp** command to copy the directory **letters** to the directory **oldnotes** on the remote system **violet**:

```
$ scp -r letters violet:oldnotes
```

NOTE *For backups or copying a large number of files, use* ***rsync****.*

At times, you may need to execute a single command on a remote system. The **rsh** command executes a Linux command on another system and displays the results on your own. Your system name and login name must, of course, be in the remote system's **.k5login** file. For SSH-enabled network connections, you can use **ssh** instead of **rsh**. The **ssh** and **rsh** commands take two general arguments: a system name and a Linux command. The syntax is as follows:

```
$ rsh remote-system-name Linux-command
```

In the next example, the **rsh** command executes an **ls** command on the remote system **violet** to list the files in the **/home/robert** directory on **violet**:

```
$ rsh violet ls /home/robert
```

Special characters are evaluated by the local system unless quoted. If you quote a special character, it becomes part of the Linux command evaluated on the remote system. Quoting redirection operators enables you to perform redirection operations on the remote system. In the next example, the redirection operator is quoted. It becomes part of the Linux command, including its argument, the filename **myfiles**. The **ls** command then generates a list of filenames that is redirected on the remote system to a file called **myfiles**, also located on the remote system.

```
$ ssh violet ls /home/robert '>' myfiles
```

The same is true for pipes. The first command (shown next) prints the list of files on the local system's printer. The standard output is piped to your own line printer. In the second command, the list of files is printed on the remote system's printer. The pipe is quoted and evaluated by the remote system, piping the standard output to the printer on the remote system.

```
$ ssh violet ls /home/robert | lpr
$ ssh violet ls /home/robert '|' lpr
```

NOTE *The Kerberos versions of the remote commands also let you specify Kerberos realms and credentials.*

V

PART

Security

CHAPTER 16
Encryption, Integrity Checks, and Signatures

CHAPTER 17
Security-Enhanced Linux

CHAPTER 18
IPsec and Virtual Private Networks

CHAPTER 19
Secure Shell and Kerberos

CHAPTER 20
Firewalls

This page intentionally left blank

CHAPTER 16

Encryption, Integrity Checks, and Signatures

You can use encryption, integrity checks, and digital signatures to protect data transmitted over a network. For example, the GNU Privacy Guard encryption package lets you encrypt your email messages or files you want to send, as well as letting you sign them with an encrypted digital signature authenticating that the message was sent by you. The digital signature also includes encrypted modification digest information that provides an integrity check, allowing the recipient to verify that the message received is the original and not one that has been changed or substituted.

This type of encryption was originally implemented with Pretty Good Privacy (PGP). Originally a privately controlled methodology, it was handed over to the Internet Engineering Task Force (IETF) to support an open standard for PGP called OpenPGP (see Table 16-1). Any project can use OpenPGP to create encryption applications, such as GnuPGP. Commercial products for PGP are still developed by the PGP Corporation, which also uses the OpenPGP standard.

Public Key Encryption, Integrity Checks, and Digital Signatures

Encrypting data is the only sure way to secure data transmitted over a network. Encrypt data with a key, and the receiver or receivers can later decrypt it. To fully protect data transmitted over a network, you should not only encrypt it, but also check that it has not been modified, as well as confirm that it was actually created by the claimed author. An encrypted message could still be intercepted and modified and then reencrypted. Integrity checks such as modification digests make sure that the data was not altered. Though encryption and integrity checks protect the data, they do not authenticate it. You also need to know that the person who claimed to send a message actually is the one who sent it, rather than an imposter. To authenticate a message, the author can sign it using a digital signature. This signature can also be encrypted, allowing the receiver to validate it. Digital signatures ensure that the message you receive is authentic.

Website	Description
gnupg.org	GnuPGP, GPG
openpgp.org	IETF open standard for PGP
pgp.com	PGP Corporation, PGP commercial products

Table 16-1 PGP Sites

Public-Key Encryption

Encryption uses a key to encrypt data in such a way that a corresponding key can decrypt it. In the past, older forms of encryption used the same key to both encrypt and decrypt a message. This, however, involved providing the receiver with the key, opening up the possibility that anyone who obtained the key could decrypt the data. Public-key encryption uses two keys to encrypt and decrypt a message, a private key and a public key. The *private* key you always keep and use to decrypt messages you have received. The *public* key you make available to those you send messages to. They then use your public key to encrypt any message they want to send to you. The private key decrypts messages, and the public key encrypts them. Each user has a set of private and public keys. Reciprocally, if you want to send messages to another user, you first obtain the user's public key and use it to encrypt the message you want to send to the user. The user then decrypts the messages with his or her own private key. In other words, your public key is used by others to encrypt the messages you receive, and you use other users' public keys to encrypt messages you send to them. All the users on your Linux system can have their own public and private keys. They will use the `gpg` program to generate them and keep their private key in their own directory.

Digital Signatures

A *digital signature* is used to both authenticate a message and provide an integrity check. Authentication guarantees that the message has not been modified—that it is the original message sent by you—and the integrity check verifies that it has not been changed. Though usually combined with encrypted messages to provide a greater level of security, digital signatures can also be used for messages that can be sent in the clear. For example, you would want to know if a public notice of upgrades of a Red Hat release was actually sent by Red Hat and not by someone trying to spread confusion. Such a message still needs to be authenticated and checked to see if it was actually sent by the sender or, if sent by the original sender, was not somehow changed en route. Verification like this protects against modification or substitution of the message by someone pretending to be the sender.

Integrity Checks

Digitally signing a message involves generating a checksum value from the contents of the message using an encryption hash algorithm such as the SHA2 modification digest algorithm. This is a unique value that accurately represents the size and contents of your message. Any changes to the message of any kind will generate a different value. Such a value provides a way to check the integrity of the data. The value is commonly known as the MD5 value,

reflective of the MD5 hash algorithm that was used encrypt the value. The MD5 algorithm has since been replaced by the more secure SHA2 algorithms.

The MD5 value is then itself encrypted with your private key. When the user receives your message, they decrypt your digital signature with your public key. The user then generates an MD5 value of the message received and compares it with the MD5 value you sent. If they are the same, the message is authenticated—it is the original message sent by you, not a false one sent by a user pretending to be you. The user can use GnuPG (described in the section "GNU Privacy Guard") to decrypt and check digital signatures.

Combining Encryption and Signatures

Normally, digital signatures are combined with encryption to provide a more secure level of transmission. The message is encrypted with the recipient's public key, and the digital signature is encrypted with your private key. The user decrypts both the message (with his or her own private key) and then the signature (with your public key). The user then compares the signature with one that user generates from the message to authenticate it. When GnuPG decodes a message, it will also decode and check a digital signature automatically. Figure 16-1 shows the process for encrypting and digitally signing a message.

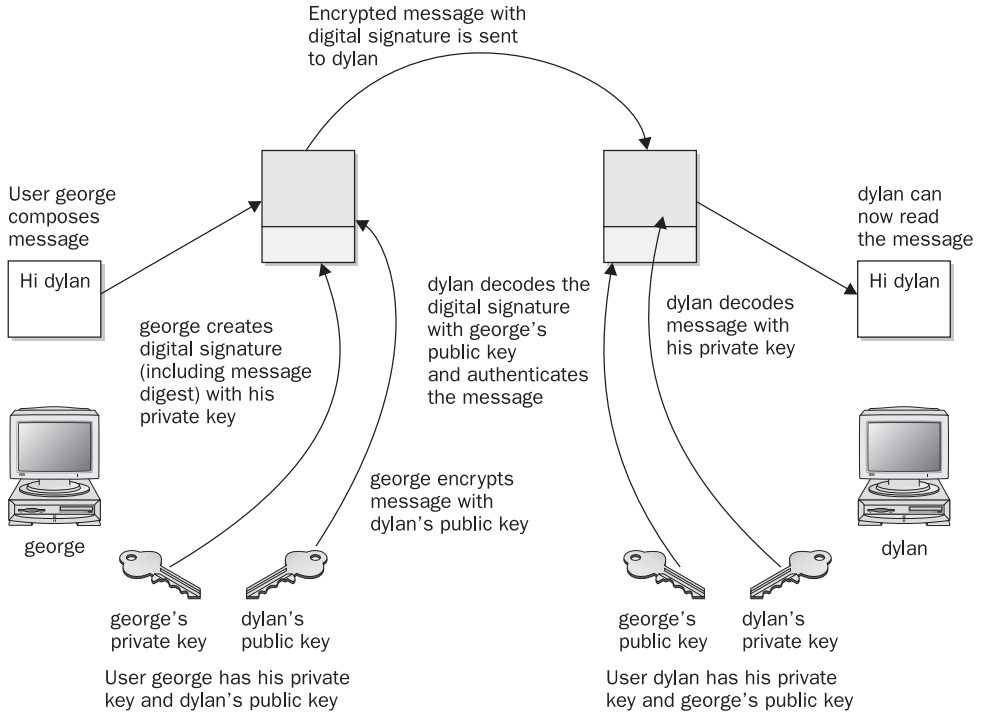


Figure 16-1 Public-key encryption and digital signatures

GNU Privacy Guard

To protect messages that you send by email, most Linux distributions provide GNU Privacy Guard (GnuPG) encryption and authentication (**gnupg.org**). GnuPG is the GNU open source software that works much like PGP encryption. It is the OpenPGP encryption and signing tool (OpenPGP is the open source version of PGP). With GnuPG, you can both encrypt your messages and digitally sign them—protecting the message and authenticating that it is from you. Currently, Evolution and KMail both support GnuPG encryption and authentication, along with Thunderbird with added GPG extensions. On Evolution, you can select PGP encryption and signatures from the Security menu to use GnuPG (the PGP options use GnuPG). On KMail, you can select the encryption to use on the Security panel in the Options window. For Thunderbird, you can use the **enigmail** extension to support OpenGPG and PGP encryption (**enigmail.mozdev.org**).

GnuPG operations are carried out with the **gpg** command, which uses both commands and options to perform tasks. Commonly used commands and options are listed in Table 16-2. Some commands and options have a short form that use only one hyphen. Normally, two hyphens are used. If you just want to verify the validity of a digital signature, you can use **gpgv** instead. This is a stripped-down version of **gpg** used just for signature verification.

GPG Commands	Description
-s, --sign	Signs a document, creating a signature. May be combined with **--encrypt**.
--clearsign	Creates a clear-text signature.
-b, --detach-sign	Creates a detached signature.
-e, --encrypt	Encrypts data. May be combined with **--sign**.
--decrypt [*file*]	Decrypts *file* (or **stdin** if no file is specified) and writes it to **stdout** (or the file specified with **--output**). If the decrypted file is signed, the signature is verified.
--verify [[*sigfile*] [*signed-files*]]	Verifies a signed file. The signature can be either contained with the file or a separate detached signature file.
--list-keys [*names*]	Lists all keys from the keyrings or those specified.
--list-public-keys [*names*]	Lists all keys from the public keyrings or those specified.
--list-secret-keys [*names*]	Lists your private (secret) keys.
--list-sigs [*names*]	Lists your keys along with any signatures they have.
--check-sigs [*names*]	Lists keys and their signatures and verifies the signatures.
--fingerprint [*names*]	Lists fingerprints for specified keys.
--gen-key	Generates a new set of private and public keys.

Table 16-2 GPG Commands and Options

GPG Commands	Description
`--edit-key` *name*	Edits your keys. Commands perform most key operations, such as `sign` to sign a key or `passwd` to change your passphrase.
`--sign-key` *name*	Signs a public key with your private key. Same as `sign in --edit-key`.
`--delete-key` *name*	Removes a public key from the public keyring.
`--delete-secret-key` *name*	Removes private and public keys from both the secret and public keyrings.
`--gen-revoke`	Generates a revocation certificate for your own key.
`--export` [*names*]	Exports a specified key from your keyring. With no arguments, exports all keys.
`--send-keys` [*names*]	Exports and sends specified keys to a keyserver. The option `--keyserver` must be used to give the name of this keyserver.
`--import` [*files*]	Imports keys contained in *files* into your public keyring.
GPG Options	**Description**
`-a, --armor`	Creates ASCII armored output, ASCII version of encrypted data.
`-o, --output` *file*	Writes output to a specified file.
`--default-key` *name*	Specifies the default private key to use for signatures.
`--keyserver` *site*	Looks up public keys not on your keyring. Can also specify the site to send your public key to. `host -l pgp.net | grep www.keys` will list the keyservers.
`-r, --recipient` *names*	Encrypts data for the specified user, using that user's public key.
`--default-recipient` *names*	Specifies the default recipient to use for encrypting data.

TABLE 16-2 GPG Commands and Options (*continued*)

The first time you use `gpg`, a **.gnugpg** directory is created in your home directory with a file named **options**. The **.gnugpg/gpg.conf** file contains commented default options for GPG operations. You can edit this file and uncomment or change any default options you want implemented for GPG. You can use a different options file by specifying it with the `--options` parameter when invoking `gpg`. Helpful options include keyserver entries. The **.gnugpg** directory will also hold encryption files such as **secring.gpg** for your secret keys (secret keyring), **pubring.gpg** for your public keys (public keyring), and **trustdb.gpg**, which is a database for trusted keys.

TIP *You can use GNOME Keyring Manager (**gnome-keyring**) to manage your PGP secret keys.*

GnuPG Setup: gpg

Before you can use GnuPG, you will have to generate your private and public keys. On the command line (terminal window), enter the **gpg** command with the **--gen-key** command. The **gpg** program will then prompt with different options for creating your private and public keys. You can check the **gpg** Man page for information on using the gpg program.

```
gpg --gen-key
```

Creating Your Key

You are first asked to select the kind of key you want. Normally, you just select the default entry, which you can do by pressing the ENTER key. Then you choose the key size, usually the default, 1024. You then specify how long the key is to be valid—usually, there is no expiration. You will be asked to enter a user ID, a comment, and an email address. Press ENTER to be prompted for each in turn. These elements, any of which can be used as the key's name, identify the key. You use the key name when performing certain GPG tasks such as signing a key or creating a revocation certificate. For example, the following elements create a key for the user richlp with the comment "author" and the email address richlp@turtle.mytrek.com:

```
Richard Petersen (author) <richlp@turtle.mytrek.com>
```

You can use any unique part of a key's identity to reference that key. For example, the string "Richard" would reference the preceding key, provided there are no other keys that have the string "Richard" in them. The string "richlp" would also reference the key, as would "author". Where a string matches more than one key, all those matched would be referenced.

Protecting Your Key

The **gpg** program will then ask you to enter a passphrase, used to protect your private key. Be sure to use a real phrase, including spaces, not just a password. **gpg** then generates your public and private keys and places them in the **.gnupg** directory. The private keys are kept in a file called **secring.gpg** in your **.gnupg** directory. The public key is placed in the **pubring.gpg** file, to which you can add the public keys of other users. You can list these keys with the **--list-keys** command.

In case you later need to change your keys, you can create a revocation certificate to notify others that the public key is no longer valid. For example, if you forget your password or someone else discovers it, you can use the revocation certificate to tell others that your public key should no longer be used. In the next example, the user creates a revocation certificate for the key richlp and places it in the file **myrevoke.asc**:

```
gpg --output myrevoke.asc --gen-revoke richlp
```

Making Your Public Key Available

For other users to decrypt your messages, you have to make your public key available to them. They, in turn, have to send you their public keys so that you can decrypt any messages you receive from them. In effect, enabling encrypted communications between users involves all of them exchanging their public keys. The public keys then have to be verified and signed by each user that receives them. The public keys can then be trusted to safely decrypt messages.

If you are sending messages to just a few users, you can manually email them your public key. For general public use, you can post your public key on a keyserver, which anyone can then download and use to decrypt any message they receive from you. A keyserver can accessed using email, LDAP, or the HTTP Horwitz Keyserver Protocol (HKP). The OpenPGP Public Keyserver project is located at **pks.sourceforge.net**. Several public keyservers are available. **hkp://subkeys.pgp.net** is listed in your **.gnupg/gpg.conf** file, though commented out. You can send directly to the keyserver with the **-keyserver** option and **--send-key** command. The **--send-key** command takes as its argument your email address. You need to send to only one keyserver, as it will share your key with other keyservers automatically.

```
gpg --keyserver search.keyserver.net --send-key chris@turtle.mytrek.com
```

If you want to send your key directly to another user, you should generate an armored text version of the key that you can then email. You do this with the **--armor** and **--export** options, using the **--output** option to specify a file to place the key in. The **--armor** option will generate an ASCII text version of the encrypted file so that it can be emailed directly, instead of as an attached binary. Files that hold an ASCII-encoded version of the encryption normally have the extension **.asc**, by convention. Binary encrypted files normally use the extension **.gpg**. You can then email the file to users you want to send encrypted messages to.

```
# gpg --armor --export richlp@turtle.mytrek.com --output richlp.asc
# mail -s 'mypubkey' george@rabbit.mytrek.com < richlp.asc
```

Many companies and institutions post their public key files on their websites, where they can be downloaded and used to verify encrypted software downloads or official announcements.

NOTE *Some commands and options for GPG have both long and short forms. For example, the* **--armor** *command can be written as* **-a**, **--output** *as* **-o**, **--sign** *as* **-s**, *and* **--encrypt** *as* **-e**. *Most others, like* **--export**, *have no short form.*

Obtaining Public Keys

To decode messages from other users, you will need to have their public keys. Either they can send them to you or you can download them from a keyserver. Save the message or web page containing the public key to a file. You will then need to import, verify, and sign the key. Use the file you received to import the public key to your **pubring** file. In the following example, the user imports George's public key, which he has received as the file **georgekey.asc**.

```
gpg --import georgekey.asc
```

All Linux distribution sites have their own public keys available for download. You should, for example, download the Red Hat public key, which can be accessed from the Red Hat site on its security resources page (**redhat.com**). Click the Public Encryption Key link. From there, you can access a page that displays just the public key. You can save this page as a file and use that file to import the Red Hat public key to your keyring. (A Red Hat distribution also places the Red Hat public key in the **/usr/share/doc/rpm4-1** directory with

versions for both GPG and PGP encryption, **RPM-GPG-KEY** and **RPM-PGP-KEY** files.) In the following example, the user saved the page showing just the Red Hat public key as **myredhat.asc**, and then imported that file:

```
gpg --import myredhat.asc
```

NOTE *You can remove any key, including your own private key, with the* ***--delete-key*** *and* ***--delete-secret-key*** *commands.*

Validating Keys

To manually check that a public key file was not modified in transit, you can check its fingerprint. This is a hash value generated from the contents of the key, much like a modification digest. Using the **--fingerprint** option, you can generate a hash value from the key you installed, and then contact the sender and ask them what the hash value should really be. If they are not the same, you know the key was tampered with in transit.

```
gpg --fingerprint george@rabbit
```

You do not have to check the fingerprint to have **gpg** operate. This is just an advisable precaution you can perform on your own. The point is that you need to be confident that the key you received is valid. Normally you can accept most keys from public servers or known sites as valid, though it is easy to check their posted fingerprints. Once assured of the key's validity, you can then sign it with your private key. Signing a key notifies **gpg** that you officially accept the key.

To sign a key, you use the **gpg** command with the **--sign-key** command and the key's name.

```
gpg --sign-key george@rabbit
```

Alternatively, you can edit the key with the **--edit-key** command to start an interactive session in which you can enter the command **sign** to sign the key and **save** to save the change. Signing a key involves accessing your private key, so you will be prompted for your passphrase. When you are finished, leave the interactive session with the **quit** command.

Normally, you will want to post a version of your public key that has been signed by one or more users. You can do the same for other users. Signing a public key provides a way to vouch for the validity of a key. It indicates that someone has already checked it out. Many different users can sign the same public key. Once you have received and verified a key from another user, you can sign and return the signed version to that user. After you sign the key, you can generate a file containing the signed public version. You can then send this file to the user. This process builds a Web of Trust, where many users vouch for the validity of public keys.

```
gpg -a --export george@rabbit --output  georgesig.asc
```

The user then imports the signed key and exports it to a keyserver.

TIP *If you want to start over from scratch, you can just erase your* ***.gnupg*** *directory, though this is a drastic measure, as you will lose any keys you have collected.*

Using GnuPG

GnuPG encryption is currently supported by most mail clients, including Kmail, Thunderbird, and Evolution. You can also use the GNU Privacy Assistant (GPA), a graphical user interface (GUI) front end, to manage GPG tasks, or you can use the **gpg** command to manually encode and decode messages, including digital signatures, if you wish. As you perform GPG tasks, you will need to reference the keys you have using their key names. Bear in mind that you need only a unique identifying substring to select the key you want. GPG performs a pattern search on the string you specify as the key name in any given command. If the string matches more than one key, all those matching will be selected. In the following example, the "Sendmail" string selects matches on the identities of two keys.

```
# gpg --list-keys "Sendmail"
pub  1024R/CC374F2D 2000-12-14
            Sendmail Signing Key/2001 <sendmail@Sendmail.ORG>
pub  1024R/E35C5635 1999-12-13
            Sendmail Signing Key/2000 <sendmail@Sendmail.ORG>
```

Encrypting Messages

The **gpg** command provides several options for managing secure messages. The **e** option encrypts messages, the **a** option generates an armored text version, and the **s** option adds a digital signature. You will need to specify the recipient's public key, which you should already have imported into your **pubring** file. It is this key that is used to encrypt the message. The recipient will then be able to decode the message with their private key. Use the **--recipient** or **-r** option to specify the name of the recipient key. You can use any unique substring in the user's public key name. The email address usually suffices. You use the **d** option to decode received messages. In the following example, the user encrypts (**e**) and signs (**s**) a file generated in armored text format (**a**). The **-r** option indicates the recipient for the message (whose public key is used to encrypt the message).

```
gpg -e -s -a -o myfile.asc -r george@rabbit.mytrek.com myfile
# mail george@rabbit.mytrek.com < myfile.asc
```

You can leave out the ASCII armor option if you want to send or transfer the file as a binary attachment. Without the **--armor** or **-a** option, **gpg** generates an encoded binary file, not an encoded text file. A binary file can be transmitted through email only as an attachment. As noted previously, ASCII armored versions usually have an extension of **.asc**, whereas binary version use **.gpg**.

NOTE *You can use* ***gpgsplit*** *to split a GPG message into its components to examine them separately.*

Decrypting Messages

When the other user receives the file, they can save it to a file named something like **myfile.asc** and then decode the file with the **-d** option. The **-o** option will specify a file to save the decoded version in. GPG will automatically determine if it is a binary file or an ASCII armored version.

```
gpg -d -o myfile.txt myfile.asc
```

To check the digital signature of the file, you use the **gpg** command with the **--verify** option. This assumes that the sender has signed the file.

```
gpg --verify myfile.asc
```

Decrypting a Digital Signature

You will need to have the signer's public key to decode and check the digital signature. If you do not, you will receive a message saying that the public key was not found. In this case, you will first have to obtain the signer's public key. You can access a keyserver that you think may have the public key or request the public key directly from a website or from the signer. Then import the key as described previously.

Signing Messages

You do not have to encrypt a file to sign it. A digital signature is a separate component. You can either combine the signature with a given file or generate one separately. To combine a signature with a file, you generate a new version that incorporates both. Use the **--sign** or **-s** option to generate a version of the document that includes the digital signature. In the following example, the **mydoc** file is digitally signed with **mydoc.gpg** file containing both the original file and the signature.

```
gpg  -o mydoc.gpg  --sign mydoc
```

If, instead, you want to just generate a separate signature file, you use the **--detach-sig** command. This has the advantage of not having to generate a complete copy of the original file. That file remains untouched. The signature file usually has an extension such as **.sig**. In the following example, the user creates a signature file called **mydoc2.sig** for the **mydoc2** file.

```
gpg -o mydoc2.sig --detach-sig mydoc2
```

To verify the file using a detached signature, the recipient user specifies both the signature file and the original file.

```
gpg --verify mydoc2.sig  mydoc2
```

To verify a trusted signature you can use **gpgv**.

You can also generate a clear sign signature to be used in text files. A *clear sign* signature is a text version of the signature that can be attached to a text file. The text file can be further edited by any text editor. Use the **--clearsign** option to create a clear sign signature. The following example creates a clear signed version of a text file called **mynotice.txt**.

```
gpg -o mysignotice.txt --clearsign mynotice.txt
```

NOTE *Numerous GUI front ends and filters are available for GnuPG at* ***www.gnupg.org.*** *GPA provides a GNOME-based front end to easily encrypt and decrypt files. You can select files to encode, choose the recipients (public keys to use), and add a digital signature, if you wish. You can also use GPA to decode encoded files you receive. You can manage your collection of public keys, the keys in your keyring file.*

TIP *Steganography is a form of encryption that hides data in other kinds of objects, such as images. You can use JPEG Hide and Seek software (JPHS) to encode and retrieve data in a JPEG image (**jphide** and **jpseek**). See **linux01.gwdg.de/~alatham/stego.html** for more details.*

Checking Software Package Digital Signatures

One very effective use for digital signatures is to verify that a software package has not been tampered with. A software package could be intercepted in transmission and some of its system-level files changed or substituted. Software packages from your distribution, as well as those by reputable GNU and Linux projects, are digitally signed. The signature provides modification digest information with which to check the integrity of the package. The digital signature may be included with the package file or posted as a separate file. You use the **gpg** command with the **--verify** option to check the digital signature for a file.

Importing Public Keys

First, however, you will need to make sure that you have the signer's public key. The digital signature was encrypted with the software distributor's private key; that distributor is the signer. Once you have that signer's public key, you can check any data you receive from them. In the case of third-party software repositories such as freshrpms.net, you will be asked to install their public key the first time you try to install any software from their site. Once the key is installed, you do not have to install it again. With Yellowdog Updater Modified (Yum), this is usually just a prompt to install the key, requesting a y or n confirmation, or it's a dialog, requesting an OK click. Repositories such as Livna and Fresprms.net will include their keys with their Yum configuration packages. You can also, if you wish, download and install them manually from their websites.

In the case of a software distributor, you can download their public key from their website or from their keyserver. Once you have their public key, you can check any software they distribute.

As noted previously, you can download the Red Hat public key from the Red Hat website security resources page or use the version installed in the Red Hat Package Manager (RPM) documentation directory. Once you have obtained the public key, you can add to your keyring with the **-import** option, specifying the name you gave to the downloaded key file (in this case, **myredhat.asc**):

```
# gpg -import redhat.asc
gpg: key CBA29BF9: public key imported
gpg: Total number processed: 1
gpg: imported: 1 (RSA: 1)
```

To download from a keyserver instead, you use the **--keyserver** option and the keyserver name.

To import the Red Hat public key from the RPM directory, you specify the file. This is the key provided by the Red Hat distribution on your DVD-ROM or CD-ROMs. Though used during installation, the key has to be imported to verify packages again after they have been installed.

```
rpm --import /usr/share/doc/rpm-4.2.2/RPM-GPG-KEY
```

Validating Public Keys

You can use the **--fingerprint** option to check a key's validity, if you wish. If you are confident that the key is valid, you can then sign it with the **--sign-key** command. In the following example, the user signs the Red Hat key, using the string "Red Hat" in the key's name to reference it. The user is also asked to enter his or her passphrase to allow use of his or her private key to sign the Red Hat public key.

```
# gpg --sign-key "Red Hat"
pub  1024R/CBA29BF9  created: 1996-02-20 expires: never  trust: -/q
(1). Red Hat Software, Inc. <redhat@redhat.com>
pub  1024R/CBA29BF9  created: 1996-02-20 expires: never  trust: -/q
 Fingerprint: 6D 9C BA DF D9 60 52 06  23 46 75 4E 73 4C FB 50
 Red Hat Software, Inc. <redhat@redhat.com>

Are you really sure that you want to sign this key
with your key: "Richard Petersen (author) <richlp@turtle.mytrek.com>"
Really sign? yes
You need a passphrase to unlock the secret key for
user: "Richard Petersen (author) <richlp@turtle.mytrek.com>"
1024-bit DSA key, ID 73F0A73C, created 2001-09-26
Enter passphrase:
#
```

Checking RPM Packages

RPM packages from any Yum or Apt repository will check the public key automatically. Should you download an RPM package separately, you can check the package manually. Once you have the software provider's public key, you can check any RPM software packages with the **rpm** command and **-K** option. The following example checks the validity of the xcdroast software packages:

```
# rpm -K xcdroast-0.98alpha9-1.i386.rpm
xcdroast-0.98alpha9-1.i386.rpm: md5 OK
```

Many software packages in the form of compressed archives, **.tar.gz** or **tar.bz2**, will provide signatures in separate files that end with the **.sig** extension. To check these, you use the **gpg** command with the **--verify** option. For example, the most recent Sendmail package is distributed in the form of a compressed archive, **.tar.gz**. Its digital signature is provided in a separate **.sig** file. First you download and install the public key for Sendmail software obtained from the Sendmail website (the key may have the year as part of its name).

```
# gpg --import sendmail2006.asc
```

You then sign the Sendmail public key that you just imported. In this example, the email address is used for the key name.

```
gpg --sign-key sendmail@Sendmail.ORG
```

You can also check the fingerprint of the key for added verification.

You then download both the compressed archive and the digital signature files. For the compressed archive (**.tar.gz**) you can use the **.sig** file ending in **.gz.sig**, and for the uncompressed archive use **.tar.sig**. Then, with the **gpg** command and the **--verify** option, use the digital signature in the **.sig** file to check the authenticity and integrity of the software compressed archive.

```
# gpg --verify sendmail.8.13.8.tar.gz.sig sendmail.8.13.8.tar.gz
gpg: Signature made Tue 08 Aug 2006 10:24:45 PM PDT using RSA key ID AF959625
gpg: Good signature from "Sendmail Signing Key/2006 <sendmail@Sendmail.ORG>"
```

You can also specify just the signature file, and **gpg** will automatically search for and select a file of the same name, but without the **.sig** or **.asc** extension.

```
# gpg --verify sendmail.8.12.0.tar.sig
```

In the future, when you download any software from the Sendmail site that uses this key, you just have to perform the **--verify** operation. Bear in mind, though, that different software packages from the same site may use different keys. You will have to make sure that you have imported and signed the appropriate key for the software you are checking.

Intrusion Detection: Tripwire and AIDE

When someone breaks into a system, they will usually try to gain control by making their own changes to system administration files, such as password files. They can create their own user and password information, allowing them access at any time, or simply change the root user password. They can also replace entire programs, such as the login program, with their own version. One method of detecting such actions is to use an integrity checking tool such as Tripwire or AIDE (Advanced Intrusion Detection Environment) to detect any changes to system administration files. AIDE is an alternative to Tripwire. It provides easy configuration and detailed reporting.

An integrity checking tool works by first creating a database of unique identifiers for each file or program to be checked. These can include features such as permissions and file size, but more importantly, they can also include checksum numbers generated by encryption algorithms from the file's contents. For example, in Tripwire, the default identifiers are checksum numbers created by algorithms such as the SHA2 modification digest algorithm and Snefru (Xerox secure hash algorithm). An encrypted value that provides such a unique identification of a file is known as a signature. In effect, a signature provides an accurate snapshot of the contents of a file. Files and programs are then periodically checked by generating their identifiers again and matching them with those in the database. Tripwire will generate signatures of the current files and programs and match them against the values previously generated for its database. Any differences are noted as changes to the file, and Tripwire then notifies you of the changes.

NOTE *You can also check your log files for any suspicious activity. The* ***/var/log/messages*** *file, in particular, is helpful for checking for critical events such as user logins, FTP connections, and superuser logins.*

Encrypted File Systems

Linux lets you encrypt nonroot and swap file systems, allowing access only to those users with the appropriate encrypted password. You can apply encryption to both fixed and removable file systems such as USB devices. It is recommended that you use the Luks (Linux Unified Key Setup) encryption tools to encrypt file systems.You can use either the **gnome-luks-format** tool or **`cryptsetup`** directly to setup your encrypted file system. If avialable for your distribution, the easiest way to set up an encrypted file system is to use the tool. This tool lets you specify the file system, the encryption cipher and passphrase, and the file system type and name. Be sure the file system is not mounted.

Once formatted, restart your system. You can then access the encrypted partition or removable drive. For a USB drive or disk, from the file system window double-click the USB drive icon. This opens a window in which you are prompted for a password with the option to forget, remember for the session, or always remember. A message tells you the device is encrypted. Once you enter your password, you can then mount and access the device (double-click it again). The volume name will appear with an icon on your desktop. HAL will handle all mounting and access for removable media. Use the same procedure for fixed partitions. Instead of restarting your system after the initialization and format, you can use **`luks-setup`** or **`crypsetup`** with the **`luksOpen`** option to open the encrypted file system. If you want to manage fixed drives manually, you can place entries in the **/etc/crypttab** and **/etc/fstab** files for them.

Instead of using **gnome-luks-format**, you can use the **`cryptsetup`** command directly to manually setup your encrypted file system. You first use the **`cryptsetup`** command with the **`luksFormat`** option to initialize and create an encrypted volume. You will be prompted to specify a key (or add the key file as an argument). Add an entry for the volume in the **/etc/crypttab** file. Then either reboot or use the **`cryptsetup`** command with the **`luksOpen`** option to access the volume. You will be prompted for the key (or use **`--keyfile`** to specify the key). You can then format the file system, specifying its name and type. Place an entry for the new file system in the **/etc/fstab** file.

If you did not use Luks, you will have to specify an encryption method with the **`cypher`** option. Use the **`--cypher`** option with **`cryptsetup`** in the **/etc/crypttab** entry. For an ESSIV cypher, you use aes-cbc-essiv:sha256. For a plain cypher, you use aes-cbc-plain.

17 CHAPTER

Security-Enhanced Linux

Though numerous security tools exist for protecting specific services, as well as user information and data, no tool has been available for protecting the entire system at the administrative level. Security-Enhanced Linux is a project to provide built-in administrative protection for aspects of your Linux system. Instead of relying on users to protect their files or on a specific network program to control access, security measures are built into the basic file management system and the network access methods. All controls can be managed directly by an administrator as part of Linux system administration.

Security-Enhanced Linux (SELinux) is a project developed and maintained by the National Security Agency (NSA), which chose Linux as its platform for implementing a secure operating system. Most Linux distributions have embraced SELinux and incorporated it as a standard feature of its distribution. Detailed documentation is available from resources listed in Table 17-1, including sites provided by the NSA and SourceForge. Also check your Linux distribution's site for any manuals, FAQs, or documentation on SELinux.

Linux and Unix systems normally use a discretionary access control (DAC) method for restricting access. In this approach, users and the objects they own, such as files, determine permissions. The user has complete discretion over the objects he or she owns. The weak point in many Linux/Unix systems has been the user administrative accounts. If an attacker managed to gain access to an administrative account, they would have complete control over the service the account managed. Access to the root user would give control over the entire system, all its users, and any network services it was running. To counter this weakness, the NSA set up a mandatory access control (MAC) structure. Instead of an all-or-nothing set of privileges based on accounts, services and administrative tasks are compartmentalized and separately controlled with policies detailing what can and cannot be done. Access is granted not just because one is an authenticated user, but when specific security criteria are met. Users, applications, processes, files, and devices can be given only the access they need to do their job, and nothing more.

Flask Architecture

The Flask architecture organizes operating system components and data into subjects and objects. Subjects are processes: applications, drivers, system tasks that are currently running. Objects are fixed components such as files, directories, sockets, network interfaces, and devices. For each subject and object, a security context is defined. A *security context* is a set of

Resource	Location
NSA SELinux	**nsa.gov/selinux**
NSA SELinux FAQ	**nsa.gov/selinux/info/faq.cfm**
SELinux at sourceforge.net	**selinux.sourceforge.net**
Writing SELinux Policy HOWTO	Accessible from "SELinux resources at sourceforge" link at **selinux.sourceforge.net**
NSA SELinux Documentation	**nsa.gov/selinux/info/docs.cfm**
Configuring SELinux Policy	Accessible from NSA SELinux Documentation
SELinux Reference Policy Project	**http://oss.tresys.com/projects/refpolicy**

Table 17-1 SELinux Resources

security attributes that determines how a subject or object can be used. This approach provides very fine-grained control over every element in the operating system as well as all data on your computer.

The attributes designated for the security contexts and the degree to which they are enforced are determined by an overall security policy. The policies are enforced by a security server. Distributions may provide different preconfigured policies from which to work. For example, Fedora provides three policies, each in its own package: strict, targeted, and mls, all a variation of a single reference policy.

SELinux uses a combination of the Type Enforcement (TE), Role Based Access Control (RBAC), and Multi-Level Security (MLS) security models. Type Enforcement focuses on objects and processes like directories and applications, whereas Role Based Access Enforcement controls user access. For the Type Enforcement model, the security attributes assigned to an object are known as either domains or types. Types are used for fixed objects such as files, and domains are used for processes such as running applications. For user access to processes and objects, SELinux makes use of the Role Based Access Control model. When new processes or objects are created, transition rules specify the type or domain they belong to in their security contexts.

With the RBAC model, users are assigned roles for which permissions are defined. The roles restrict what objects and processes a user can access. The security context for processes will include a role attribute, controlling what objects it can assess. The new Multi-Level Security (MLS) adds a security level, containing both a sensitivity and capability value.

Users are given separate SELinux user identities. Normally these correspond to the user IDs set up under the standard Linux user creation operations. Though they may have the same name, they are not the same identifiers. Standard Linux identities can be easily changed with commands like `setuid` and `su`. Changes to the Linux user ID will not affect the SELinux ID. This means that even if a user changes his or her ID, SELinux will still be able to track it, maintaining control over that user.

System Administration Access

It is critically important that you make sure you have system administrative access under SELinux before you enforce its policies. This is especially true if you are using a strict or mls policy, which imposes restrictions on administrative access. You should always use SELinux

in permissive mode first and check for any messages denying access. With SELinux enforced, it may no longer matter whether you can access the root user. What matters is whether your user, even the root user, has sysadm_r role and sysadm_t object access and an administrative security level. You may not be able to use the **su** command to access the root user and expect to have root user administrative access. Recall that SELinux keeps its own security identities that are not the same as Linux user IDs. Though you might change your user ID with **su**, you still have not changed your security ID.

The targeted policy will set up rules that allow standard system administrator access using normal Linux procedures. The root user will be able to access the root user account normally. In the strict policy, however, the root user needs to access its account using the appropriate security ID. Both are now part of a single reference policy. If you want administrative access through the **su** command (from another user), you first use the **su** command to log in as the root user. You then have to change your role to that of the sysadm_r role, and you must already be configured by SELinux policy rules to be allowed to take on the sysadm_r role. A user can have several allowed possible roles he or she can assume.

To change the role, you use the **newrole** command with the **-r** option.

```
newrole -r sysadm_r
```

Terminology

SELinux uses several terms that have different meanings in other contexts. The terminology can be confusing because some of the terms, such as *domain*, have different meanings in other, related, areas. For example, a domain in SELinux is a process as opposed to an object, whereas in networking the term refers to network DNS addresses.

Identity

SELinux creates identities with which to control access. Identities are not the same as traditional user IDs. At the same time, each user normally has an SELinux identity, though the two are not linked. Affecting a user does not affect the corresponding SELinux identity. SELinux can set up a separate corresponding identity for each user, though on the less secure policies, such as targeted policies, general identities are used. A general user identity is used for all normal users, restricting users to user-level access, whereas administrators are given administrative identities. You can further define security identities for particular users.

The identity makes up part of a security context that determines what a user can or cannot do. Should a user change user IDs, that user's security identity will not change. A user will always have the same security identity. In traditional Linux systems, a user can use commands like **su** to change his or her user ID, becoming a different user. On SELinux, even though a user can still change his or her Linux user ID, the user still retains the same original security ID. You always know what a particular person is doing on your system, no matter what user ID that person may assume.

The security identity can have limited access. So, even though a user may use the Linux **su** command to become the root user, the user's security identity could prevent him or her from performing any root user administrative commands. As noted previously, to gain an administrative access, the role for their security identity would have to change as well.

Use **id -Z** to see what the security context for your security identity is, what roles you have, and what kind of objects you can access. This will list the user security context that starts with the security ID, followed by a colon, and then the roles a user has and the objects the user can control. Security identities can have roles that control what they can do. A user role is user_r, and a system administration role is system_r. The general security identity is user_u, whereas a particular security identity will normally use the username. The following example shows a standard user with the general security identity:

```
$ id -Z
 user_u:user_r:user_t
```

In this example the user has a security identity called george:

```
$ id -Z
 george:user_r:user_t
```

You can use the **newrole** command to change the role a user is allowed. Changing to a system administrative role, the user can then have equivalent root access.

```
$ id -Z
 george:sysadm_r:sysadm_t
```

Domains

Domains are used to identify and control processes. Each process is assigned a domain within which it can run. A domain sets restrictions on what a process can do. Traditionally, a process was given a user ID to determine what it could do, and many had to have root user ID to gain access to the full file system. This also could be used to gain full administrative access over the entire system. A domain, on the other hand, can be tailored to access some areas but not others. Attempts to break into another domain, such as the administrative domain, would be blocked. For example, the administrative domain is sysadm_t, whereas the DNS server uses only named_t, and users have a user_t domain.

Types

Whereas domains control processes, *types* control objects like files and directories. Files and directories are grouped into types that can be used to control who can have access to them. The type names have the same format as the domain names, ending with a _t suffix. Unlike domains, types reference objects, including files, devices, and network interfaces.

Roles

Types and domains are assigned to roles. Users (security identities) with a given role can access types and domains assigned to that role. For example, most users can access user_t type objects but not sysadm_t objects. The types and domains a user can access are set by the role entry in configuration files. The following example allows users to access objects with the user password type:

```
role user_r types user_passwd_t
```

Security Context

Each object has a security context that set its security attributes. These include identity, role, domain or type. A file will have a security context listing the kind of identity that can access it, the role under which it can be accessed, and the security type it belongs to. Each component adds its own refined level of security. Passive objects are usually assigned a generic role, object_r, which has no effect, as such objects cannot initiate actions.

A normal file created by users in their own directories will have the following identity, role, and type. The identity is a user and the role is that of an object. The type is the user's home directory. This type is used for all subdirectories and their files created within a user's home directory.

```
user_u:object_r:user_home_t
```

A file or directory created by that same user in a different part of the file system will have a different type. For example, the type for files created in the **/tmp** directory will be tmp_t.

```
user_u:object_r:tmp_t
```

Transition: Labeling

A *transition,* also known as labeling, assigns a security context to a process or file. For a file, the security context is assigned when it is created, whereas for a process the security context is determined when the process is run.

Making sure every file has an appropriate security context is called *labeling.* Adding another file system requires that you label (add security contexts) to the directories and files on it. Labeling varies, depending on the policy you use. Each policy may have different security contexts for objects and processes. Relabeling is carried out using the **`fixfiles`** command in the policy source directory.

```
fixfiles relabel
```

Policies

A *policy* is a set of rules to determine the relationships between users, roles, and types or domains. These rules state what types a role can access and what roles a user can have.

Multi-Level Security (MLS) and Multi-Category Security (MCS)

Multi-Level Security (MLS) adds a more refined security access method, designed for servers. MLS adds a security level value to resources. Only users with access to certain levels can access the corresponding files and applications. Within each level, access can be further controlled with the use of categories. Categories work much like groups, allowing access only to users cleared for that category. Access becomes more refined, instead of an all-or-nothing situation.

Multi-Category Security (MCS) extends SELinux to use not only by administrators, but also by users. Users can set categories that can restrict and control access to their files and applications. Though based on MLS, it uses only categories, not security levels. Users can select a category for a file, but only the administrator can create a category and determine

what users can access it. Though similar in concept to an ACL (Access Control List), it differs in that it makes use of the SELinux security structure, providing user-level control enforced by SELinux.

Management Operations for SELinux

Certain basic operations, such as checking the SELinux status, checking a user's or file's security context, or disabling SELinux at boot, can be very useful.

Turning Off SELinux

Should you want to turn off SELinux before you even start up your system, you can do so at the boot prompt. Just add the following parameter to the end of your GRUB boot line:

```
selinux=0
```

To turn off SELinux permanently, set the **SELINUX** variable in the **/etc/selinux/config** file to **disabled**:

```
SELINUX=disabled
```

To turn off (permissive mode) SELinux temporarily without rebooting, use the **setenforce** command with the **0** option; use **1** to turn it back on (enforcing mode). You can also use the terms **permissive** or **enforcing** at the arguments instead of **0** or **1**. You must first have the sysadm_r role, which you can obtain by logging in as the root user.

```
setenforce 1
```

Checking Status and Statistics

To check the current status of your SELinux system, use **sestatus**. Adding the **-v** option will also display process and file contexts, as listed in **/etc/sestatus.conf**. The contexts will specify the roles and types assigned to a particular process, file, or directory.

```
sestatus -v
```

Use the **seinfo** command to display your current SELinux statistics:

```
# seinfo
Statistics for policy file: /etc/selinux/targeted/policy/policy.21
Policy Version & Type: v.21 (binary, MLS)

   Classes:            55    Permissions:       206
   Types:            1043    Attributes:         85
   Users:               3    Roles:               6
   Booleans:          135    Cond. Expr.:       138
   Sensitivities:       1    Categories:        256
   Allow:           46050    Neverallow:          0
   Auditallow:         97    Dontaudit:        3465
   Role allow:          5    Role trans:          0
   Type_trans:        987    Type_change:        14
```

```
Type_member:          0    Range_trans:         10
Constraints:          0    Validatetrans:        0
Fs_use:              12    Genfscon:            52
Portcon:            190    Netifcon:             0
Nodecon:              8    Initial SIDs:         0
```

Checking Security Context

The **-Z** option used with the **ls**, **id**, and **ps** commands can be used to check the security context for files, users, and processes respectively. The security context tells you the roles that users must have to access given processes or objects.

```
ls -lZ
id -Z
ps -eZ
```

SELinux Management Tools

SELinux provides a number of tools to let you manage your SELinux configuration and policy implementation, including semanage to configure your policy. The setools collection provides SELinux configuration and analysis tools including **apol**, the Security Policy Analysis tool for domain transition analysis, **sediffx** for policy differences, and **seaudit** to examine the auditd logs (see Table 17-2). The command line user management tools, **useradd**, **usermod**, and **userdel**, all have SELinux options that can be applied when SELinux is installed. In addition, the **audit2allow** tool will convert SELinux denial messages into policy modules that will allow access.

Command	Description
seinfo	Displays policy statistics.
sestatus	Checks status of SELinux on your system, including the contexts of processes and files.
sesearch	Searches for Type Enforcement rules in policies.
seaudit	Examines SELinux log files.
sediffx	Examines SELinux policy differences.
autid2allow	Generates policy to allow rules for modules using audit AVC denial messages.
apol	The SELinux Policy Analysis tool.
checkpolicy	The SELinux policy compiler.
fixfiles	Checks file systems and sets security contexts.
restorecon	Sets security features for particular files.
newrole	Assigns new role.
setfiles	Sets security context for files.
chcon	Changes context.
chsid	Changes security ID.

TABLE 17-2 SELinux Tools

With the modular version of SELinux, policy management is no longer handled by editing configuration files directly. Instead, you use the SELinux management tools such as the command line tool **semanage**. Such tools make use of interface files to generate changed policies.

semanage

semanage lets you change your SELinux configuration without having to edit SELinux source files directly. It covers several major categories including users, ports, file contexts, and logins. Check the Man page for **semanage** for detailed descriptions. Options let you modify specific security features such as **-s** for the username, **-R** for the role, **-t** for the type, and
-r for an MLS security range. The following example adds a user with role user_r.

```
semanage user -a -R user_r  justin
```

semanage is configured with the **/etc/selinux/semanage.conf** file, where you can set semanage to write directly on modules (the default) or work on the source.

The Security Policy Analysis Tool: apol

The SELinux Policy Analysis tool, **apol**, provides a complex and detailed analysis of a selected policy. Select the **apol** entry in the Administration menu to start it.

Checking SELinux Messages: seaudit

SELinux AVC messages are now saved in the **/var/log/audit/audit.log** file. These are particularly important if you are using the permissive mode to test a policy you want to later enforce. You need to find out if you are being denied access when appropriate, and afforded control when needed. To see only the SELinux messages, you can use the seaudit tool. Startup messages for the SELinux service are still logged in **/var/log/messages**.

Allowing Access: chcon and audit2allow

Whenever SELinux denies access to a file or application, the kernel issues an AVC notice. In many cases the problem can be fixed simply by renaming the security context of a file to allow access. You use the **chcon** command to change a file's security context. In this rename, access needs to be granted to the Samba server for a **log.richard3** file in the **/var/lib/samba** directory.

```
chcon -R -t samba_share_t log.richard3
```

More complicated problems, especially ones that are unknown, may require you to create a new policy module using the AVC messages in the audit log. To do this, you can use the **audit2allow** command. This command will take an audit AVC messages and generate commands to allow SELinux access. The audit log is **/var/log/autid/audit.log**. This log is outputted to **audit2allow**, which then can use its **-M** option to create a policy module.

```
cat /var/log/audit/audit.log | audit2allow -M local
```

You then use the semodule command to load the module:

```
semodule -i local.pp
```

If you want to first edit the allowable entries, you can use the following to create a **.te** file of the local module, **local.te**, which you can then edit:

```
audit2allow -m local -i  /var/log/audit/audit.log   >  local.te
```

Once you have edited the **.te** file, you can use **checkmodule** to compile the module, and then **semodule_package** to create the policy module, **local.pp**. Then you can install it with **semodule**. You first create a **.mod** file with **checkmodule**, and then a **.pp** file with **semodule_package**.

```
checkmodule -M -m -o local.mod local.te
semodule_package -o local.pp  -m local.mod


semodule -i local.pp
```

In this example the policy module is called **local**. If you later want to create a new module with **audit2allow**, you should either use a different name or append the output to the **.te** file using the **-o** option.

TIP *On Red Hat and Fedora distributions, you can use the SELinux Troubleshooter to detect SELinux access problems.*

The SELinux Reference Policy

A system is secured using a policy. SELinux now uses a single policy, the reference policy, instead of the two separate targeted and strict policies used in previous editions (see **serefpolicy.sourceforge.net**). Instead of giving users just two alternatives, strict and targeted, the SELinux reference policy project aims to provide a basic policy that can be easily adapted and expanded as needed. The SELinux reference policy configures SELinux into modules that can be handled separately. You still have strict and targeted policies, but they are variations on a basic reference policy. In addition, you can have an MLS policy for Multi-Level Security. The targeted policy is installed by default, and you can install the strict or MLS policies yourself.

On some distributions, such as Fedora, there may be separate policy configurations already provided. For example, Fedora currently provides three effective policies : targeted, strict, and mls. The targeted policy is used to control specific services, like network and Internet servers such as web, DNS, and FTP servers. It also can control local services with network connections. The policy will not affect just the daemon itself, but all the resources it uses on your system.

The strict policy provides complete control over your system. It is under this kind of policy that your users, and even administrators, can be inadvertently locked out of the system. A strict policy needs to be carefully tested to make sure access is denied or granted when appropriate.

There will be **targeted**, **strict**, and **mls** subdirectories in your **/etc/selinux** directory, but they now each contain a **modules** directory. It is here that you will find your SELinux configurations.

Multi-Level Security (MLS)

Multi-Level Security (MLS) add a more refined security access method. MLS adds a security level value to resources. Only users with access to certain levels can access the corresponding files and applications. Within each level, access can be further controlled with the use of categories. Categories work much like groups, allowing access only to users cleared for that category. Access becomes more refined, instead of an all-or-nothing situation.

Multi-Category Security (MCS)

Multi-Category Security (MCS) extends SELinux to use not only by administrators, but also by users. Users can set categories that restrict and control access to their files and applications. Though based on MLS, MCS uses only categories, not security levels. Users can select a category for a file, but only the administrator can create a category and determine what users can access it. Though similar in concept to an ACL (access control list), it differs in that it makes use of the SELinux security structure, providing user-level control enforced by SELinux.

Policy Methods

Operating system services and components are categorized in SELinux by their type and their role. Rules controlling these objects can be type based or role based. Policies are implemented using two different kinds of rules, Type Enforcement (TE) and Role Based Access Control (RBAC). Multi-Level Security (MLS) is an additional method further restricting access by security level. Security context now features both the role of an object, such as a user, and that object's security level.

Type Enforcement

With a type structure, the operating system resources are partitioned off into types, with each object assigned a type. Processes are assigned to domains. Users are restricted to certain domains and allowed to use only objects accessible in those domains.

Role-Based Access Control

A role-based approach focuses on controlling users. Users are assigned roles that define what resources they can use. In a standard system, file permissions, such as those for groups, can control user access to files and directories. With roles, permissions become more flexible and refined. Certain users can have more access to services than others.

SELinux Users

Users will retain the permissions available on a standard system. In addition, SELinux can set up its own controls for a given user, defining a role for that user. General security identities created by SELinux include:

- **system_u** The user for system processes
- **user_u** To allow normal users to use a service
- **root** For the root user

Policy Files

Policies are implemented in policy files. These are binary files compiled from source files. For a preconfigured targeted policy file, the policy binary files are in policy subdirectories in the **/etc/selinux** configuration directory, **/etc/selinux/targeted**. For example, the policy file for the targeted policy is

```
/etc/selinux/targeted/policy/policy.20
```

The targeted development files that hold the interface files are installed at **/usr/share/selinux**.

```
/usr/share/selinux/targeted
```

You can use the development files to create your own policy modules that you can then load.

SELinux Configuration

Configuration for general SELinux server settings is carried out in the **/etc/selinux/config** file. Currently there are only two settings to make: the state and the policy. You set the SELINUX variable to the state, such as enforcing or permissive, and the SELINUXTYPE variable to the kind of policy you want. These correspond to the Securitylevel-config SELinux settings for disabled and enforcing, as well as the policy to use, such as targeted (the targeted name may be slightly different on different distributions, like **refpolicy-targeted** used on Debian). A sample config file is shown here:

```
# This file controls the state of SELinux on the system.
# SELINUX= can take one of these three values:
#     enforcing - SELinux security policy is enforced.
#     permissive - SELinux prints warnings instead of enforcing.
#     disabled - SELinux is fully disabled.
SELINUX=permissive
# SELINUXTYPE= type of policy in use. Possible values are:
#     targeted - Only targeted network daemons are protected.
#     strict - Full SELinux protection.
SELINUXTYPE=targeted
```

SELinux Policy Rules

Policy rules can be made up of either type (Type Enforcement, or TE) or RBAC (Role Based Access Control) statements, along with security levels (Multi-Level Security). A type statement can be a type or attribute declaration or a transition, change, or assertion rule. The RBAC statements can be role declarations or dominance, or they can allow roles. A security level specifies a number corresponding to the level of access permitted. Policy configuration can be difficult, using extensive and complicated rules. For this reason, many rules are implemented using M4 macros in **fi** files that will in turn generate the appropriate rules (Sendmail uses M4 macros in a similar way). You will find these rules in files in the SELinux reference policy source code package that you need to download and install.

Type and Role Declarations

A type declaration starts with the keyword **type**, followed by the type name (identifier) and any optional attributes or aliases. The type name will have a _t suffix. Standard type definitions are included for objects such as files. The following is a default type for any file, with attributes file_type and sysadmfile:

```
type file_t, file_type, sysadmfile;
```

The root will have its own type declaration:

```
type root_t, file_type, sysadmfile;
```

Specialized directories such as the boot directory will also have their own type:

```
type boot_t, file_type, sysadmfile;
```

More specialized rules are set up for specific targets like the Amanda server. The following example is the general type definition for amanda_t objects, those objects used by the Amanda backup server, as listed in the targeted policy's **src/program/amanda.te** file:

```
type amanda_t, domain, privlog, auth, nscd_client_domain;
```

A role declaration determines the roles that can access objects of a certain type. These rules begin with the keyword **role** followed by the role and the objects associated with that role. In this example, the amanda objects (amanda_t) can be accessed by a user or process with the system role (system_r):

```
role system_r types amanda_t;
```

A more specific type declaration is provided for executables, such as the following for the Amanda server (amanda_exec_t). This defines the Amanda executable as a system administration–controlled executable file.

```
type amanda_exec_t, file_type, sysadmfile, exec_type;
```

Associated configuration files often have their own rules:

```
type amanda_config_t, file_type, sysadmfile;
```

In the targeted policy, a general unconfined type is created that user and system roles can access, giving complete unrestricted access to the system. More specific rules will restrict access to certain targets like the web server.

```
type unconfined_t, domain, privuser, privhome, privrole, privowner, admin,
auth_write, fs_domain, privmem;
role system_r types unconfined_t;
role user_r types unconfined_t;
role sysadm_r types unconfined_t;
```

Types are also set up for the files created in the user home directory:

```
type user_home_t, file_type, sysadmfile, home_type;
type user_home_dir_t, file_type, sysadmfile, home_dir_type;
```

File Contexts

File contexts associate specific files with security contexts. The file or files are listed first, with multiple files represented with regular expressions. Then the role, type, and security level are specified. The following creates a security context for all files in the **/etc** directory (configuration files). These are accessible from the system user (system_u) and are objects of the etc_t type with a security level of 0, s0.

```
/etc(/.*)?              system_u:object_r:etc_t:s0
```

Certain files can belong to other types; for instance, the **resolve.conf** configuration file belongs to the net_conf type:

```
/etc/resolv\.conf.*     --      system_u:object_r:net_conf_t:s0
```

Certain services will have their own security contexts for their configuration files:

```
/etc/amanda(/.*)?            system_u:object_r:amanda_config_t:s0
```

File contexts are located in the **file_contexts** file in the policy's contexts directory, such as **/etc/selinux/targeted/contexts/files/file_contexts**. The version used to create or modify the policy is located in the policy modules active directory, as in **targeted/modules/active/file_contexts**.

User Roles

User roles define what roles a user can take on. Such a role begins with the keyword **`user`** followed by the username, then the keyword **`roles`**, and finally the roles it can use. You will find these rules in the SELinux reference policy source code files. The following example is a definition of the system_u user:

```
user system_u roles system_r;
```

If a user can have several roles, then they are listed in brackets. The following is the definition of the standard user role in the targeted policy, which allows users to take on system administrative roles:

```
user user_u roles { user_r sysadm_r system_r };
```

The strict policy lists only the user_r role:

```
user user_u roles { user_r };
```

Access Vector Rules: allow

Access vector rules are used to define permissions for objects and processes. The **`allow`** keyword is followed by the object or process type and then the types it can access or be accessed by and the permissions used. The following allows processes in the amanda_t domain to search the Amanda configuration directories (any directories of type amanda_config_t):

```
allow amanda_t amanda_config_t:dir search;
```

The following example allows Amanda to read the files in a user home directory:

```
allow amanda_t user_home_type:file { getattr read };
```

The next example allows Amanda to read, search, and write files in the Amanda data directories:

```
allow amanda_t amanda_data_t:dir { read search write };
```

Role Allow Rules

Roles can also have allow rules. Though they can be used for domains and objects, they are usually used to control role transitions, specifying whether a role can transition to another role. These rules are listed in the RBAC configuration file. The following entry allows the user to transition to a system administrator role:

```
allow user_r sysadm_r;
```

Transition and Vector Rule Macros

The type transition rules set the type used for rules to create objects. Transition rules also require corresponding access vector rules to enable permissions for the objects or processes. Instead of creating separate rules, macros are used that will generate the needed rules. The following example sets the transition and access rules for user files in the home directory, using the file_type_auto_trans macro:

```
file_type_auto_trans(privhome, user_home_dir_t, user_home_t)
```

The next example sets the Amanda process transition and acce ss rules for creating processes:

```
domain_auto_trans(inetd_t, amanda_inetd_exec_t, amanda_t)
```

Constraint Rules

Restrictions can be further placed on processes such as transitions to ensure greater security. These are implemented with constraint definitions in the constraints file. Constraint rules are often applied to transition operations, such as requiring that, in a process transition, user identities remain the same, or that process 1 be in a domain that has the privuser attribute and process 2 be in a domain with the userdomain attribute. The characters u, t, and r refer to user, type, and role, respectively.

```
constrain process transition
    ( u1 == u2 or ( t1 == privuser and t2 == userdomain )
```

SELinux Policy Configuration Files

Configuration files are normally changed using **.te** and **.fc** files. These are missing from the module headers in **/usr/share/selinux**. If you are adding a module you will need to create the **.te** and **.fc** files for it. Then you can create a module and add it as described in the next section. If you want to create or modify your own policy, you will need to download and

install the source code files for the SELinux reference policy, as described the section after "**Using SELinux Source Configuration**". The reference policy code holds the complete set of **.te** and **.fc** configuration files.

Compiling SELinux Modules

Instead of compiling the entire source each time you want to make a change, you can just compile a module for the area you changed. The modules directory holds the different modules. Each module is built from a corresponding **.te** file. The **checkmodule** command is used to create a **.mod** module file from the **.te** file, and then the **semodule_package** command is used to create the loadable **.pp** module file as well as a **.fc** file context file. As noted in the SELinux documentation, if you need to change the configuration for **syslogd**, you first use the following to create a **syslogd.mod** file using **syslogd.te**. The **-M** option specifies support for MLS security levels.

```
checkmodule -M -m syslogd.te  -o syslogd.mod
```

Then use the **semodule_package** command to create a **syslogd.pp** file from the **syslogd .mod** file. The **-f** option specifies the file context file.

```
semodule_package -m syslogd.mod  -o syslogd.pp -f syslogd.fc
```

To add the module you use **semodule** and the **-i** option. You can check if a module is loaded with the **-l** option.

```
semodule -i syslogd.pp
```

Changes to the base policy are made to the **policy.conf** file, which is compiled into the **base.pp** module.

Using SELinux Source Configuration

To perform you own configuration, you will have to download and install the source code file for the SELinux reference policy. For RPM distributions, this will be an SRPMS file. The **.te** files used for configuring SELinux are no longer part of the SELinux binary packages.

NOTE *On Red Hat and Fedora distributions, the compressed archive of the source, a tgz file, along with various policy configuration files, will be installed to* ***/usr/src/redhat/SOURCES****. (Be sure you have already installed* ***rpm-build****; it is not installed by default.) You use an* ***rpmbuild*** *operation with the* ***security-policy.spec*** *file to extract the file to the* ***serefpolicy*** *directory in* ***/usr/src/redhat/BUILD****.*

Change to the **seref-policy** directory and run the following command to install the SELinux source to **/etc/selinux/serefpolicy/src**.

```
make install-src
```

The rules are held in configuration files located in various subdirectories in a policy's **src** directory. Within this directory you will find a **policy/modules** subdirectory. There, organized into several directories, such as **admin** and **apps**, you will find the **.tc**, **.fc**, and **.if** configuration files.

You will have configuration files for both Type Enforcement and security contexts. Type Enforcement files have the extension **.te**, whereas security contexts have an **.sc** extension.

Reflecting the fine-grained control that SELinux provides, you have numerous module configuration files for the many kinds of objects and processes on your system. The primary configuration files and directories are listed in Table 17-3, but several expand to detailed listing of subdirectories and files.

Interface Files

File *interface* files allow management tools to generate policy modules. They define interface macros for your current policy. The **refpolicy** SELinux source file will hold **.if** files for each module, along with **.te** and **.fc** files. Also, the **.if** files in the **/usr/share/selinux/devel** directory can be used to generate modules.

Directories and Files	Description
assert.te	Access vector assertions
config/appconfig-*	Application runtime configuration files
policy/booleans.conf	Tunable features
file_contexts	Security contexts for files and directories
policy/flask	Flask configuration
policy/mcs	Multi-Category Security (MCS) configuration
doc	Policy documentation support
policy/modules	Security policy modules
policy/modules.conf	Module list and use
policy/modules/admin	Administration modules
policy/modules/apps	Application modules
policy/modules/kernel	Kernel modules
policy/modules/services	Services and server modules
policy/modules/system	System modules
policy/rolemap	User domain types and roles
policy/users	General users definition
config/local.users	Your own SELinux users
policy/constraints	Additional constraints for role transition and object access
policygentool	Script to generate policies
policy/global_tunables	Policy tunables for customization
policy/mls	Multi-Level Security (MLS) configuration

Table 17-3 SELinux Policy Configuration Files

Types Files

In the targeted policy, the modules directory that defines types holds a range of files, including **nfs.te** and **network.te** configuration files. Here you will find type declarations for the different kinds of objects on your system. The **.te** files are no longer included with your standard SELinux installation. Instead, you have to download and install the serefpolicy source package. This is the original source and allows you to completely reconfigure your SELinux policy, instead of managing modules with management tools like **semanage**. The modules directory will hold **.te** files for each module, listing their TE rules.

Module Files

Module are located among several directories in the **policy/modules** directory. Here you will find three corresponding files for each application or service. There will be a **.te** file that contains the actual Type Enforcement rules, an **.if**, for interface (a file that allows other applications to interact with the module), and the **.fc** files that define the file contexts.

Security Context Files

Security contexts for different files are detailed in security context files. The **file_contexts** file holds security context configurations for different groups, directories, and files. Each configuration file has an **.fc** extension. The **types.fc** file holds security contexts for various system files and directories, particularly access to configuration files in the **/etc** directory. In the SELinux source, each module will have its own **.fc** file, along with corresponding **.te** and **.if** files. The **distros.fc** file defines distribution-dependent configurations. The **homedir_template** file defines security contexts for dot files that may be set up in a user's home directory, such as **.mozilla**, **.gconf**, and **.java**.

A modules directory has file context files for particular applications and services. For example, **apache.fc** has the security contexts for all the files and directories used by the Apache web server, such as **/var/www** and **/etc/httpd**.

User Configuration: Roles

Global user configuration is defined in the policy directory's **users** file. Here you find the user definitions and the roles they have for standard users (user_u) and administrators (admin_u). To add your own users, you use the **local.users** file. Here you will find examples for entering your own SELinux users. Both the strict and targeted policies use the general user_u SELinux identity for users. To set up a separate SELinux identity for a user, you define that user in the **local.users** file.

The **rbac** file defines the allowed roles one role can transition to. For example, can the user role transition to an system administration role? The targeted policy has several entries allowing a user to freely transition to an administrator, and vice versa. The strict policy has no such definitions.

Role transitions are further restricted by rules in the **constraints** file. Here the change to other users is controlled, and changing object security contexts (labeling) is restricted.

Policy Module Tools

To create a policy module and load it, you use several policy module tools. First the **checkmodule** command is used to create **.mod** file from a **.te** file. Then the **semodule_package** tool takes the **.mod** file and any supporting **.fc** file, and generates a module policy

package file, **.pp**. Finally, the **semodule** tool can take the policy package file and install it as part of your SELinux policy.

Application Configuration: appconfig

Certain services and applications are security aware and will request default security contexts and types from SELinux (see also the upcoming section "Runtime Security Contexts and Types: contexts"). The configuration is kept in files located in the **policy/config/appconfig**-* directory. The **default_types** file holds type defaults; **default_contexts** holds default security contexts. The **initrc_context** file has the default security context for running **/etc/rc.d** scripts. A special **root_default_contexts** file details how the root user can be accessed. The **removable_context** file holds the security context for removable devices, and **media** lists media devices, such as cdrom for CD-ROMs. Runtime values can also be entered in corresponding files in the policy contexts directory, such as **/etc/selinux/targeted/contexts**.

Creating an SELinux Policy: make and checkpolicy

If you want to create an entirely new policy, you use the SELinux reference policy source, **/etc/selinux/serefpolicy**. Once you have configured your policy, you can create it with the **`make policy`** and **`checkpolicy`** commands. The **`make policy`** command generates a **policy.conf** file for your configuration files, which **`checkpolicy`** can then use to generate a policy binary file. A policy binary file will be created in the **policy** subdirectory with a numeric extension for the policy version, such as **policy.20**.

You will have to generate a new **policy.conf** file. To do this you enter the following command in the policy src directory, which will be **/etc/selinux/serefpolicy/src/policy**.

```
make policy
```

Then you can use **`checkpolicy`** to create the new policy.

Instead of compiling the entire source each time you want to make a change, you can just compile a module for the area you changed. (In the previous SELinux version, you always had to recompile the entire policy every time you made a change.) The modules directory holds the different modules. Each module is built from a corresponding **.te** file. The **`checkmodule`** command is used to create a **.mod** module file from the **.te** file, and then the **`semanage_module`** command is used to create the loadable policy package **.pp** module file. As noted in the SELinux documentation, if you need to just change the configuration for **syslogd**, you would first use the following to create a **syslogd.mod** file using **syslogd.te**. The **`-M`** option specifies support for MLS security levels.

```
checkmodule -M  -m syslogd.te  -o syslogd.mod
```

Then use the **`semanage_module`** command to create a **syslogd.pp** file from the syslogd .mod file. The **`-f`** option specifies the file context file.

```
semanage_module -m syslogd.mod  -o syslogd.pp -f syslogd.fc
```

To add the module, you use **`semodule`** and the **`-i`** option. You can check if a module is loaded with the **`-l`** option.

```
semodule -i syslogd.pp
```

Changes to the base policy are made to the **policy.conf** file, which is compiled into the **base.pp** module.

To perform your own configuration, you will now have to download the source code files. The **.te** files used for configuring SELinux are no longer part of the SELinux binary packages. Once installed, the source will be in the **sefepolicy** directory in **/etc/selinux**.

SELinux: Administrative Operations

There are several tasks you can perform on your SELinux system without having to recompile your entire configuration. Security contexts for certain files and directories can be changed as needed. For example, when you add a new file system, you will need to label it with the appropriate security contexts. Also, when you add users, you may need to have a user be given special attention by the system.

Using Security Contexts: fixfiles, setfiles, restorecon, and chcon

Several tools are available for changing your objects' security contexts. The **fixfiles** command can set the security context for file systems. You use the **relabel** option to set security contexts and the **check** option to see what should be changed. The **fixfiles** tool is a script that uses **setfiles** and **restorecon** to make actual changes.

The **restorecon** command will let you restore the security context for files and directories, but **setfiles** is the basic tool for setting security contexts. It can be applied to individual files or directories. It is used to label the file when a policy is first installed.

With **chcon**, you can change the permissions of individual files and directories, much as **chmod** does for general permissions.

Adding New Users

If a new user needs no special access, you can generally just use the generic SELinux user_u identity. If, however, you need to allow the user to take on roles that would otherwise be restricted, such as a system administrator role in the strict policy, you need to configure the user accordingly. To do this, you add the user to the **local.users** file in the policy users directory, as in **/etc/selinux/targeted/policy/users/local.users**. Note that this is different from the **local.users** file in the **src** directory, which is compiled directly into the policy. The user rules have the syntax

```
user username roles { rolelist };
```

The following example adds the sysadm role to the george user:

```
user george roles { user_r sysadm_r };
```

Once the role is added, you have to reload the policy.

```
make reload
```

You can also manage users with the **semanage** command with the **user** option. To see what users are currently active, you can list them with the **semanage user** command and the **-l** option.

```
# semanage user -l
system_u: system_r
user_u: user_r sysadm_r system_r
root: user_r sysadm_r system_r
```

The **semanage user** command has **a**, **d**, **m**, options for adding, removing, or changing users, respectively. The **a** and **m** options let you specify roles to add to a user, whereas the **d** option will remove the user.

Runtime Security Contexts and Types: contexts

Several applications and services are security aware, and will need default security configuration information such as security contexts. Runtime configurations for default security contexts and types are kept in files located in the policy context directory, such as **/etc/selinux/targeted/contexts**. Types files will have the suffix **_types**, and security context files will use **_context**. For example, the default security context for removable files is located in the **removable_context** file. The contents of that file are shown here:

```
system_u:object_r:removable_t
```

The **default_context** file is used to assign a default security context for applications. In the strict policy it is used to control system admin access, providing it where needed, for instance, during the login process.

The following example sets the default roles for users in the login process:

```
system_r:local_login_t user_r:user_t
```

This allows users to log in either as administrators or as regular users.

```
system_r:local_login_t sysadm_r:sysadm_t user_r:user_t
```

This next example is for remote user logins, for which system administration is not included:

```
system_r:remote_login_t user_r:user_t staff_r:staff_t
```

The **default_types** file defines default types for roles. This files has role/type entries, and when a transition takes place to a new role, the default type specified here is used. For example, the default type for the sysadm_r role is sysadm_t.

```
sysadm_r:sysadm_t
user_r:user_t
```

Of particular interest is the **initrc_context** file, which sets the context for running the system scripts in the **/etc/rc.d** directory. In the targeted policy these are open to all users.

```
user_u:system_r:unconfined_t
```

In the strict policy these are limited to the system user.

```
system_u:system_r:initrc_t
```

users

Default security contexts may also need to be set up for particular users such as the root user. In the **sesusers** file you will find a root entry that lists roles, types, and security levels the root user can take on, such as the following example for the su operation (on some distributions this may be a **users** directory with separate files for different users):

```
sysadm_r:sysadm_su_t  sysadm_r:sysadm_t staff_r:staff_t user_r:user_t
```

context/files

Default security contexts for your files and directories are located in the **contexts/files** directory. The **file_contexts** directory lists the default security contexts for all your files and directories, as set up by your policy. The **file_context.homedirs** directory sets the file contexts for user home directory files as well as the root directory, including dot configuration files like **.mozilla** and **.gconf**. The media file sets the default context for media devices such as CD-ROMs and disks.

```
cdrom system_u:object_r:removable_device_t
floppy system_u:object_r:removable_device_t
disk system_u:object_r:fixed_disk_device_t
```

This page intentionally left blank

CHAPTER 18

IPsec and Virtual Private Networks

The Internet Security Protocol, IPsec, incorporates security for network transmission into the Internet Protocol (IP) directly. IPsec is integrated into the new IPv6 protocol (Internet Protocol, version 6). It can also be used with the older IPv4 protocol. IPsec provides methods for both encrypting data and authenticating the host or network it is sent to. The process can be handled manually or automated using the IPsec `racoon` key exchange tool. With IPsec, the kernel can automatically detect and decrypt incoming transmissions, as well as encrypt outgoing ones. You can also use IPsec to implement virtual private networks, encrypting data sent over the Internet from one local network to another. Though IPsec is a relatively new security method, its integration into the Internet Protocol will eventually provide it wide acceptance. Check the IPsec HOWTO for a detailed explanation of IPsec implementation on Linux, **ipsec-howto.org**.

Several projects currently provide development and implementation of IPsec tools (see Table 18-1). The original IPsec tools are provided by the KAME project, **kame.net**. Current versions can be obtained from **ipsec-tools.souceforge.net**. Other IPsec tool projects include the Open Secure/Wide Area Network project (Openswan) at **openswan.org**, which provides a Linux implementation of IPsec tools, and the VPN Consortium (VPNC) at **vpnc.org**, which supports Windows and Macintosh versions. Documentation will be located at **/usr/doc/openswan-***version*. Detailed documentation is held in the **openswan-doc** package, which will be installed at **/usr/doc/openswan-doc-***version*.

IPsec Protocols

IPsec is made up of several protocols that provide authentication, encryption, and the secure exchange of encryption keys. The Authentication Header protocol (AH) confirms that the packet was sent by the sender and not by someone else. IPsec also includes an integrity check to detect any tampering in transit. Packets are encrypted using the Encapsulating Security Payload (ESP). Encryption and decryption are performed using secret keys shared by the sender and the receiver. These keys are themselves transmitted using the Internet Key Exchange (IKE) protocol, which provides a secure exchange. ESP encryption can degrade certain compression transmission methods, such as PPP for dial-up

Website	Project
kame.net	KAME project for IPsec tools
openswan.org	Open Secure/Wide Area Network project
vpnc.org	VPN Consortium
ipsec-howto.org	IPsec HOWTO documentation
ipsec-tools.sourceforge.net	IPsec tools and resources

TABLE 18-1 IPsec Resources

Internet connections. To accommodate these compression methods, IPsec provides the IP Payload Compression Protocol (IPComp), with which packets can be compressed before being sent.

Encrypted authentication and integrity checks are included using Hash Methods Authentication Codes (HMAC) generated from hash security methods like SHA2 using a secret key. The HMAC is included in the IPsec header, which the receiver can then check with the secret key. Encryption of transmitted data is performed by symmetric encryption methods like 3DES, Blowfish, and DES.

The AH, ESP, and IPComp protocols are incorporated into the Linux kernel. The IKE protocol is implemented as a separate daemon. It simply provides a way to share secret keys and can be replaced by other sharing methods.

IPsec Modes

You can use IPsec capabilities for either normal transport or packet tunneling. With normal transport, packets are encrypted and sent to the next destination. The normal transport mode is used to implement direct host-to-host encryption, where each host handles the IPsec encryption process. Packet tunneling is used to encrypt transmissions between gateways, letting the gateways handle the IPsec encryption process for traffic directed to or from an entire network, rather than having to configure IPsec encryption for each host. With packet tunneling, the packets are encapsulated with new headers for a specific destination, enabling you to implement virtual private networks (VPNs). Packets are directed to VPN gateways, which encrypt and send on local network packets.

NOTE *You can choose to encrypt packets for certain hosts or for those passing through specific ports.*

IPsec Security Databases

The packets you choose to encrypt are designated by the IPsec Security Policy Database (SPD). The method you use to encrypt them is determined by the IPsec Security Association Database (SAD). The SAD associates an encryption method and key with a particular connection or kind of connection. The connections to be encrypted are designated in the Security Policy Database.

Tool	Description
`plainrsa-gen`	Generates a plain RSA key.
`setkey`	Manages policy (SPD) and association (SAD) databases.
`raccoon`	Configures and implements secure key exchanges using IPsec Key Exchange (IKE).
`racoonctl`	Administers IPsec connections.

TABLE 18-2 IPsec Tools

IPsec Tools

Several IPsec tools are provided with which you can manage your IPsec connections (see Table 18-2). With **`setkey`**, you can manage both the policy and association databases. The **`racoon`** tool configures the key exchange process to implement secure decryption key exchanges across connections. To administer your IPsec connections, you can use **`racoonctl`**. For example, the **`show-sa`** option will display your security associations, and the **`vpn-connect`** will establish a VPN connection.

NOTE *To enable IPsec in the kernel, be sure to enable the PF_KEY, AH, and ESP options in Cryptographic Options.*

Configuring Connections with setkey

To configure your IPsec connections, you can use the **`setkey`** tool. This tool contains several instructions for managing rules in the IPsec policy and security databases. You use the **`add`** instruction to add a security association to the security database (SAD) and the **`spdadd`** instruction to add a policy to the policy database (SPD). The **`ah`** term designates that the instruction is being applied to the authentication header (AH), and **`esp`** indicates the encryption is to be implemented by the encryption security payload (ESP). To implement **`setkey`** operations, it is best to use a script invoking **`setkey`** with the **`-f`** option and listing the **`setkey`** instructions. The following example creates a simple script to add authentication and encryption instructions for a particular connection, as well as create a security policy for it:

```
#!/sbin/setkey -f
add 192.168.0.2 192.168.0.5 ah 15700 -A hmac-md5 "secret key";
add 192.168.0.2 192.168.0.5  esp 15701 -E 3des-cbc "secret key ";
spdadd 192.168.0.2 192.168.0.5  any -P out ipsec
   esp/transport//require
   ah/transport//require;
```

Security Associations: SA

You use security associations to indicate you want the authentication header (AH) and encryption payload (ESP) encrypted. A particular connection, such as that between two hosts, can have those hosts' authentication headers encrypted using specified encryption methods

and designated secret keys. The same can be done for the encryption payload, the main content of transmissions. A secret key can be determined manually or automatically using key exchanges. The following example specifies that for the connection between 192.168.0.2 and 192.168.0.5, the **hmac-md5** authentication method and a secret key (here designated by the placeholder **secret key**) will be used for the authentication header, **ah**.

```
add 192.168.0.2 192.168.0.5 ah 15700 -A hmac-md5 "secret key";
```

The security association for the encryption payload uses the 3des-cbc encryption method and a different secret key.

```
add 192.168.0.2 192.168.0.5 esp 15701 -E 3des-cbc "secret key";
```

Each instruction is identified with a security parameter index (SPI), in this case, 15700 and 15701. In fact, identical instructions with different SPIs are considered different instructions.

Bear in mind that the security associations only specify possible encryption procedures. They do not implement them. For that, you need to set security policies.

Security Policy: SP

A security policy will implement an IPsec security procedure for a connection. You can designate a host or port connection. Once a policy is set for a connection, the kernel will determine what security associations to apply, using the SAD database. A security policy is added with the **spdadd** instruction. Either encryption, authentication, or both can be required.

The following example will encrypt and authenticate transmissions between hosts 192.168.0.2 and 192.168.0.5. Any outgoing transmissions between these hosts will be both encrypted and authenticated:

```
spdadd 192.168.0.2 192.168.0.5 any -P out ipsec esp/transport//require
ah/transport/require;
```

In the **spdadd** instruction, you will need to specify the connection, such as one between two hosts or two networks. For two hosts, you use their IP addresses, in this example, 192.168.0.2 and 192.168.0.5. You then specify the kind of packet and its direction, in this case any outgoing packet, **any -P out**. Then you can specify the **ipsec** directives for either the ESP or AH protocol, or both. For each entry, you specify the mode (transport or tunnel), the hosts involved (this can be different in tunnel mode), and the policy for the encryption, usually **require**. This example shows that the ESP protocol will use the transport mode for connections between 192.168.02 and 192.168.0.5, and it will be required:

```
esp/transport/192.168.02-192.168.0.5/require
```

You can leave out the host information if it is the same, as in the prior example.

```
esp/transport//require
```

Receiving Hosts

For a host to receive an encrypted IPsec transmission, it must have corresponding security association instructions in its own SAD database that tell it how to authenticate and decrypt the received instructions. The security association instructions mirror those of the

sender's instructions, using the same encryption method, secret keys, and security indexes. A corresponding policy, though, is not required.

```
#!/sbin/setkey -f
add 192.168.0.2 192.168.0.5 ah 15700 -A hmac-md5 "secret key";
add 192.168.0.2 192.168.0.5 esp 15701 -E 3des-cbc "secret key";
```

Receiving hosts may want to set up policies to screen incoming packets on secure connections, discarding those not encrypted. The following policy will accept only incoming IPsec encrypted and authenticated transmissions from 192.168.0.2.

```
spdadd 192.168.0.2 192.168.0.5 any -P in ipsec esp/transport//require
ah/transport//require;
```

Two-Way Transmissions

The preceding example set up a secure connection between two hosts going only one way, from 192.168.0.2 to 192.168.0.5, not the other way, from 192.168.0.5 to 192.168.0.2. To implement two-way secure transmissions between two hosts, both need to be configured as the sender and the receiver, with corresponding security associations to match. The following scripts are based on common examples of a simple two-way IPsec connection between two hosts. They set up a secure two-way IPsec connection between hosts 192.168.0.2 and 192.168.0.5. Corresponding incoming policies are also included but not required.

First is the configuration for host 192.168.0.2:

```
#!/sbin/setkey -f
add 192.168.0.2 192.168.0.5 ah 15700 -A hmac-md5 "secret key";
add 192.168.0.5 192.168.0.2 ah 24500 -A hmac-md5 "secret key";

add 192.168.0.2 192.168.0.5 esp 15701 -E 3des-cbc "secret key";
add 192.168.0.5 192.168.0.2 esp 24501 -E 3des-cbc "secret key";

spdadd 192.168.0.2 192.168.0.5 any -P out ipsec esp/transport//require
ah/transport//require;
spdadd 192.168.0.5 192.168.0.2 any -P in ipsec esp/transport//require
ah/transport//require;
```

The corresponding host, 192.168.0.5, uses the same instructions but with the IP connections reversed. Notice that the security indexes for instructions for the sender and receiver at each end correspond:

```
#!/sbin/setkey -f
add 192.168.0.5 192.168.0.2 ah 15700 -A hmac-md5 "secret key";
add 192.168.0.2 192.168.0.5 ah 24500 -A hmac-md5 "secret key";

add 192.168.0.5 192.168.0.2 esp 15701 -E 3des-cbc "secret key";
add 192.168.0.2 192.168.0.5 esp 24501 -E 3des-cbc "secret key";

spdadd 192.168.0.5 192.168.0.2 any -P out ipsec esp/transport//require ah/
transport//require;
spdadd 192.168.0.2 192.168.0.5 any -P in ipsec esp/transport//require
ah/transport//require;
```

Configuring IPsec with racoon: IKE

IPsec keys can be implemented as manual keys, as shared keys, or with certificates. Manual keys are explicitly exchanged and are prone to security problems. Both shared keys and certificates are managed using the IPsec Key Exchange protocol, which will automatically exchange keys, changing them randomly to avoid detection.

One of the advantages of using IKE is that it will automatically generate any needed security associations if none are provided. This means that to configure secure connections with IKE you need to specify only a security policy, not the security associations.

The **racoon** tool is the key exchange daemon for the IPsec IKE protocol. In the case of shared keys, hosts are authenticated dynamically by **racoon** using preshared secret keys. With the certificate method, hosts are authenticated using certificate files. The **racoon** configuration file is located at **/etc/racoon/racoon.conf**. Here you can set general parameters. You can use the default **racoon.conf** file for most connections.

The **racoon** configuration consists of stanzas containing parameters for possible connections. A very simple configuration is shown in the following example, which uses a simple shared secret key. The location is specified by the **path pre_shared_key** option, in this case **/etc/racoon/psk.txt**. Certificate keys, a more secure method using public and private keys, are discussed later.

```
path pre_shared_key "/etc/racoon/psk.txt";

remote anonymous
{
      exchange_mode aggressive,main;
      doi ipsec_doi;
      situation identity_only;

      my_identifier address;

      lifetime time 2 min;   # sec,min,hour
      initial_contact on;
      proposal_check obey;    # obey, strict or claim

      proposal {
            encryption_algorithm 3des;
            hash_algorithm sha1;
            authentication_method pre_shared_key;
            dh_group 2 ;
      }
}
sainfo anonymous
{
      pfs_group 1;
      lifetime time 2 min;
      encryption_algorithm 3des, blowfish, des, cast128, rijndael ;
      authentication_algorithm hmac_sha1, hmac_md5;
            compression_algorithm deflate ;
}
```

This configuration defines stanzas for default (anonymous) connections. The **remote anonymous** stanza defines parameters for connecting to remote systems, and the **sainfo anonymous** section provides information for security association instructions, such as the encryption and authentication methods to use.

Certificates

To use certificates instead of shared keys, you first have to create certificates using OpenSSL. Then instruct **racoon** to use them. Specify the path for the certificates.

```
path certificate "/usr/local/etc/racoon/certs";
```

You can now configure **racoon** to use the public and private keys generated by the certificates. In the appropriate stanza in the **/etc/racoon/racoon.conf** file, the **certificate_type** directive specifies the public and private keys for this system. The **peers_certfile** directive specifies the location of the remote system's public key. The **authentication_method** directive is now set to **rsasig**, the RSA public/private keys. Make sure each system has its corresponding public and private keys.

```
certificate_type x509 "192.168.0.2.public" "192.168.0.2.private";
peers_certfile "192.168.0.5.public";
authentication_method rsasig;
```

Connection Configuration with racoon

With **racoon**, you will only need to specify the security policy for the connection configuration, as shown here for the sender. The receiver will have corresponding policies:

```
spdadd 192.168.0.5 192.168.0.2 any -P out ipsec
           esp/transport//require
           ah/transport//require;
spdadd 192.168.0.2 192.168.0.5 any -P in ipsec
           esp/transport//require
           ah/transport//require;
```

IPsec and IP Tables: Net Traversal

IPtables netfiltering will stop many IPsec packets. To enable IPtables to pass IPsec packets, use the following IPtables commands. The number for the AH protocol is 51, and for the ESP protocol, it is 50. To allow IPsec packets, you should set policy rules such as the following:

```
iptables -A INPUT -p 50 -j ACCEPT
iptables -A OUTPUT -p 51 -j ACCEPT
```

For netfiltering that implements IP masquerading, you will need to add a **net_traversal** option to your raccoon IPsec configuration. With Net Traversal, the IPsec connection will bypass the IP address substitution performed by IP Tables when masquerading IP addresses. In addition, the **nat_keepalive** option will maintain the connection, and with the **iskamp_natt** option, you specify the IP address and port to connect to.

IPsec Tunnel Mode: Virtual Private Networks

Instead of encrypting two hosts directly, you can use IPsec to just encrypt the gateways between the networks those hosts belong to, assuming that communication within those networks can be trusted. This significantly reduces the encryption configuration setup, letting hosts from an entire network reach those of another network, using an intermediate secure IPsec connection between their gateways. For connections between gateways, transmissions sent through intervening routers can be tunneled. This is known as the tunnel mode for IPsec, which is used to implement virtual private networks (VPNs). Encrypted transmissions between gateways effectively implement a VPN, securing transmissions across a larger network from one local net to another.

To tunnel transmissions from a host through a gateway to a network, you use the **`-m tunnel`** option. The IPsec connection is between the two gateways. The following example is the security association on gateway 10.0.0.1 that encrypts transmissions from gateway 10.0.0.1 to gateway 10.0.23.5. The examples used here are for a gateway-to-gateway connection, set up as a direct connection between two hosts using manual keys.

```
add 10.0.0.1 10.0.23.5 esp 34501 -m tunnel -E 3des-cbc "secretkey";
```

The security policy on 10.0.0.1 then implements encryption for communication from one network to another using their respective gateways. The two networks are 192.168.0.0 and 192.168.1.0. Transmissions from hosts on the 192.168.0.0 network are encrypted by their gateway, 10.0.0.1, and are then sent to the gateway for the 192.168.1.0 network, 10.0.23.5, which then decrypts them.

```
spdadd 192.168.0.0/24 192.168.1.0/24 any -P out ipsec esp/tunnel/10.0.0.1-
10.0.23.5/require;
```

Notice that the gateway IP addresses are specified in the **`spdadd`** instruction's **`ipsec`** directive. The mode specified is the tunnel mode, rather than the transport mode.

```
ipsec esp/tunnel/10.0.0.1-10.0.23.5/require
```

The receiving gateway, 10.0.23.5, will have a corresponding security association and policy, as shown here. The policy is set for incoming transmissions. In both gateway configurations, other than specifying the tunnel option and using network addresses in the security policy, the security associations and policies are the same as those used for host-to-host connections.

```
add 10.0.0.1 10.0.23.5 esp 34501 -m tunnel -E 3des-cbc "secretkey";

spdadd 192.168.0.0/24 192.168.1.0/16 any -P in ipsec esp/tunnel/10.0.0.1-
10.0.23.5/require;
```

To set up full two-way communication, the two gateways have corresponding security associations and policies to handle traffic in both directions. The following example is for the configuration on gateway 10.0.0.1 and handles two-way traffic to and from gateway 10.0.23.5. Gateway 10.0.23.5 has a similar configuration:

```
add 10.0.0.1 10.0.23.5 esp 34501 -m tunnel -E 3des-cbc "secretkey";
add 10.0.23.5 10.0.03.1 esp 34501 -m tunnel -E 3des-cbc "secretkey";
```

```
spdadd 192.168.0.0/24 192.168.1.0/24 any -P out ipsec esp/tunnel/10.0.0.1-
10.0.23.5/require;

spdadd 192.168.1.0/16 192.168.0.0/24 any -P in ipsec esp/tunnel/10.0.23.5-
10.0.0.1/require;
```

If you use **racoon** to configure gateway connections, you have to set only the security policies on each gateway, letting the **racoon** server generate the needed security associations.

This page intentionally left blank

CHAPTER 19

Secure Shell and Kerberos

To protect remote connections from hosts outside your network, transmissions can be encrypted (see Table 19-1). For Linux systems, you can use the Secure Shell (SSH) suite of programs to encrypt and authenticate transmissions, preventing them from being read or modified by anyone else, as well confirming the identity of the sender. The SSH programs are meant to replace remote tools such as rsh and rcp, which perform no encryption and include security risks such as transmitting passwords in clear text. User authentication can be controlled for certain services by Kerberos servers. Kerberos authentication provides another level of security whereby individual services can be protected, allowing use of a service only to users who are cleared for access.

The Secure Shell: OpenSSH

Although a firewall can protect a network from attempts to break into it from the outside, the problem of securing legitimate communications to the network from outside sources still exists. A particular problem is one of users who want to connect to your network remotely. Such connections could be monitored, and information such as passwords and user IDs used when the user logs in to your network could be copied and used later to break in. One solution is to use SSH for remote logins and other kinds of remote connections such as FTP transfers. SSH encrypts any communications between the remote user and a system on your network.

Two different implementations of SSH currently use what are, in effect, two different and incompatible protocols. The first version of SSH, known as SSH1, uses the original SSH protocol. Version 2.0, known as SSH2, uses a completely rewritten version of the SSH protocol. Encryption is performed in different ways, encrypting different parts of a packet. SSH1 uses server and host keys to authenticate systems, whereas SSH2 uses only host keys. Furthermore, certain functions, such as sftp, are supported only by SSH2.

NOTE *A commercial version of SSH is available from SSH Communications Security, whose website is* ***ssh.com.*** *SSH Communications Security provides an entirely commercial version called SSH Tectia, designed for enterprise and government use. The older noncommercial SSH package is still freely available, which you can download and use.*

Website	Description
openssh.org	OpenSSH open source version of SSH
ssh.com	SSH Communications Security, commercial SSH version
web.mit.edu/kerberos	Kerberos authentication

Table 19-1 SSH and Kerberos Resources

The SSH protocol has become an official Internet Engineering Task Force (IETF) standard. A free and open source version is developed and maintained by the OpenSSH project, currently supported by the OpenBSD project. OpenSSH is the version supplied with most Linux distributions, including Fedora, Red Hat, Novell, and Debian. You can find out more about OpenSSH at **openssh.org**, where you can download the most recent version, though your distribution will provide current RPM versions.

SSH Encryption and Authentication

SSH secures connections by both authenticating users and encrypting their transmissions. The authentication process is handled with public-key encryption. Once authenticated, transmissions are encrypted by a cipher agreed upon by the SSH server and client for use in a particular session. SSH supports multiple ciphers. Authentication is applied to both hosts and users. SSH first authenticates a particular host, verifying that it is a valid SSH host that can be securely communicated with. Then the user is authenticated, verifying that the user is who they say they are.

SSH uses strong encryption methods, and their export from the United States may be restricted. Currently, SSH can deal with the following kinds of attacks:

- IP spoofing, where a remote host sends out packets that pretend to come from another, trusted host
- IP source routing, where a host can pretend an IP packet comes from another, trusted host
- DNS spoofing, where an attacker forges name server records
- Interception of clear-text passwords and other data by intermediate hosts
- Manipulation of data by people in control of intermediate hosts
- Attacks based on listening to X authentication data and spoofed connections to the X11 server

Encryption

The public key encryption used in SSH authentication makes use of two keys: a public key and a private key. The *public key* is used to encrypt data, and the *private key* decrypts it. Each host or user has its own public and private keys. The public key is distributed to other hosts, who can then use it to encrypt authentication data that only the host's private key can decrypt. For example, when a host sends data to a user on another system, the host encrypts the authentication data with a public key, which it previously received from that user. The data can be decrypted only by the user's corresponding private key. The public key can

safely be sent in the open from one host to another, allowing it to be installed safely on different hosts. You can think of the process as taking place between a client and a server. When the client sends data to the server, it first encrypts the data using the server's public key. The server can then decrypt the data using its own private key.

It is recommended that SSH transmissions be authenticated with public-private keys controlled by passphrases. Unlike PGP, SSH uses public-key encryption for the authentication process only. Once authenticated, participants agree on a common cipher to use to encrypt transmissions. Authentication will verify the identity of the participants. Each user who intends to use SSH to access a remote account first needs to create the public and private keys along with a passphrase to use for the authentication process. A user then sends the public key to the remote account they want to access and installs the public key on that account. When the user attempts to access the remote account, that account can then use the users' public key to authenticate that the user is who they claim to be. The process assumes that the remote account has set up its own SSH private and public key. For the user to access the remote account, they will have know the remote account's SSH passphrase. SSH is often used in situations where a user has two or more accounts located on different systems and wants to be able to securely access them from each other. In that case, the user already has access to each account and can install SSH on each, giving each its own private and public keys along with their passphrases.

Authentication

The mechanics of authentication in SSH version 1 and version 2 differ slightly. However, the procedure on the part of users is the same. Essentially, a user creates both public and private keys. For this you use the **`ssh-keygen`** command. The user's public key then has to be distributed to those users that the original user wants access to. Often this is an account a user has on another host. A passphrase further protects access. The original user will need to know the other user's passphrase to access it.

SSH version 1 uses RSA authentication. When a remote user tries to log in to an account, that account is checked to see if it has the remote user's public key. That public key is then used to encrypt a challenge (usually a random number) that can be decrypted only by the remote user's private key. When the remote user receives the encrypted challenge, that user decrypts the challenge with its private key. SSH version 2 can use either RSA or DSA authentication. The remote user will first encrypt a session identifier using its private key, signing it. The encrypted session identifier is then decrypted by the account using the remote user's public key. The session identifier has been previously set up by SSH for that session.

SSH authentication is first carried out with the host, and then with users. Each host has its own host keys, public and private keys used for authentication. Once the host is authenticated, the user is queried. Each user has his or her own public and private keys. Users on an SSH server who want to receive connections from remote users will have to keep a list of those remote user's public keys. Similarly, an SSH host will maintain a list of public keys for other SSH hosts.

SSH Tools

SSH is implemented on Linux systems with OpenSSH. The full set of OpenSSH packages includes the general OpenSSH package (openssh), the OpenSSH server (openssh-server), and the OpenSSH clients (openssh-clients). These packages also require OpenSSL (openssl), which installs the cryptographic libraries that SSH uses.

Application	Description
Ssh	SSH client
Sshd	SSH server (daemon)
Sftp	SSH FTP client, Secure File Transfer Program. Version 2 only. Use **?** to list sftp commands (SFTP protocol)
sftp-server	SSH FTP server. Version 2 only (SFTP protocol)
Scp	SSH copy command client
ssh-keygen	Utility for generating keys. **-h** for help
ssh-keyscan	Tool to automatically gather public host keys to generate ssh_known_ hosts files
ssh-add	Adds RSD and DSA identities to the authentication agent
ssh-agent	SSH authentication agent that holds private keys for public key authentication (RSA, DSA)
ssh-askpass	X Window System utility for querying passwords, invoked by **ssh-add** (openssh-askpass)
ssh-askpass-gnome	GNOME utility for querying passwords, invoked by **ssh-add**
ssh-signer	Signs host-based authentication packets. Version 2 only. Must be suid root (performed by installation)
Slogin	Remote login (version 1)

TABLE 19-2 SSH Tools

The SSH tools are listed in Table 19-2. They include several client programs such as scp and ssh, as well as the ssh server. The ssh server (sshd) provides secure connections to anyone from the outside using the ssh client to connect. Several configuration utilities are also included, such as ssh-add, which adds valid hosts to the authentication agent, and ssh-keygen, which generates the keys used for encryption.

For version 2, names of the actual tools have a *2* suffix. Version 1 tools have a *1* as their suffix. During installation, however, links are set for each tool to use only the name with the suffix. For example, if you have installed version 2, there is a link called scp to the scp2 application. You can then use the link to invoke the tool. Using scp starts scp2. Table 19-2 specifies only the link names, as these are the same for each version. Remember, though, some applications, such as sftp, are available only with version 2.

SSH Setup

Using SSH involves creating your own public and private keys and then distributing your public key to other users you want to access. These can be different users or simply user accounts of your own that you have on remote systems. Often people remotely log in from a local client to an account on a remote server, perhaps from a home computer to a company computer. Your home computer would be your client account, and the account on your company computer would be your server account. On your client account, you need to

generate your public and private keys and then place a copy of your public key in the server account. You can do this by simply e-mailing the key file or copying the file from a floppy disk. Once the account on your server has a copy of your client user's public key, you can access the server account from your client account. You will be also prompted for the server account's passphrase. You will have to know this to access that account. Figure 19-1 illustrates the SSH setup that allows a user **george** to access the account **cecelia**.

To allow you to use SSH to access other accounts:

- You must create public and private keys on your account along with a passphrase. You will need to use this passphrase to access your account from another account.
- You must distribute your public key to other accounts you want to access, placing them in the **.ssh/authorized_keys** file.
- Other accounts also have to set up public and private keys along with a passphrase.
- You must know the other account's passphrase to access it.

Creating SSH Keys with ssh-keygen

You create your public and private keys using the **`ssh-keygen`** command. You need to specify the kind of encryption you want to use. You can use either DSA or RSA encryption. Specify the type using the **`-t`** option and the encryption name in lowercase (**`dsa`** or **`rsa`**). In the following example, the user creates a key with the RSA encryption:

```
ssh-keygen -t rsa
```

The **`ssh-keygen`** command prompts you for a passphrase, which it will use as a kind of password to protect your private key. The passphrase should be several words long. You are also prompted to enter a filename for the keys. If you do not enter one, SSH will use its defaults. The public key will be given the extension **.pub**. The **`ssh-keygen`** command generates the public key and places it in your **.ssh/id_dsa.pub** or **.ssh/id_rsa.pub** file,

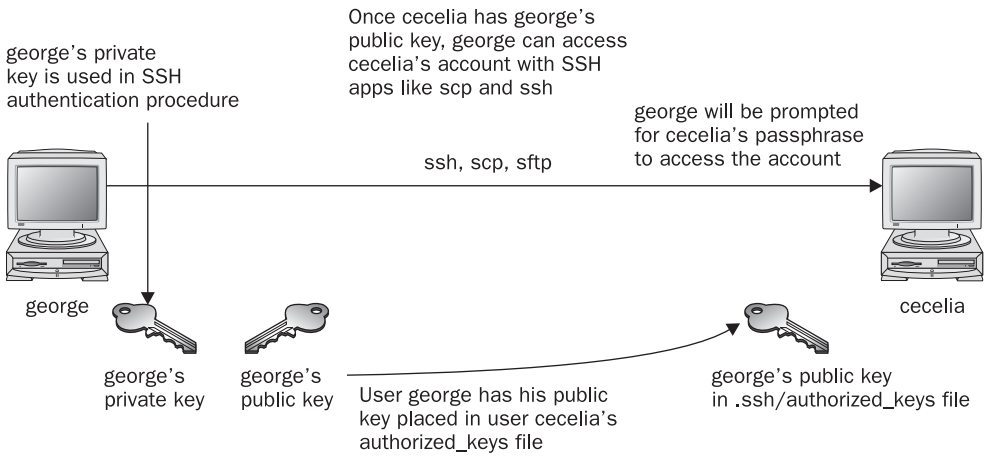


Figure 19-1 SSH setup and access

depending on the type of key you specified; it places the private key in the corresponding **.ssh/id_dsa** or **.ssh/id_rsa.pub** This is file.

NOTE *The **.ssh/identity** filename is used in SSH version 1; it may be installed by default on older distribution versions. SSH version 2 uses a different filename, **.ssh/id_dsa** or **.ssh/id_rsa,** depending on whether RSA or DSA authentication is used.*

If you need to change your passphrase, you can do so with the **`ssh-keygen`** command and the **`-p`** option. Each user will have their own SSH configuration directory, called **.ssh**, located in their own home directory. The public and private keys, as well as SSH configuration files, are placed here. If you build from the source code, the **`make install`** operation will automatically run **`ssh-keygen`**. Table 19-3 lists the SSH configuration files.

File	Description
$HOME/.ssh/known_hosts	Records host keys for all hosts the user has logged in to (that are not in **/etc/ssh/ssh_known_hosts**).
$HOME/.ssh/random_seed	Seeds the random number generator.
$HOME/.ssh/id_rsa	Contains the RSA authentication identity of the user.
$HOME/.ssh/ id_dsa	Contains the DSA authentication identity of the user.
$HOME/.ssh/id_rsa.pub	Contains the RSA public key for authentication. The contents of this file should be added to **$HOME/.ssh/authorized_keys** on all machines where you want to log in using RSA authentication.
$HOME/.ssh/id_dsa.pub	Contains the DSA public key for authentication. The contents of this file should be added to **$HOME/.ssh/authorized_keys** on all machines where you want to log in using DSA authentication.
$HOME/.ssh/config	The per-user configuration file.
$HOME/.ssh/authorized_keys	Lists the RSA or DSA keys that can be used for logging in as this user.
/etc/ssh/ssh_known_hosts	Contains the systemwide list of known host keys.
/etc/ssh/ssh_config	Contains the systemwide configuration file. This file provides defaults for those values not specified in the user's configuration file.
/etc/ssh/sshd_config	Contains the SSH server configuration file.
/etc/ssh/sshrc	Contains the system default. Commands in this file are executed by ssh when the user logs in, just before the user's shell (or command) is started.
$HOME/.ssh/rc	Contains commands executed by ssh when the user logs in, just before the user's shell (or command) is started.

TABLE 19-3 SSH Configuration Files

Authorized Keys

A public key is used to authenticate a user and its host. You use the public key on a remote system to allow that user access. The public key is placed in the remote user account's **.ssh/authorized_keys** file. Recall that the public key for DSA is held in the **.ssh/id_dsa.pub** file. If a user wants to log in remotely from a local account to an account on a remote system, they would first place their public key for DSA in the **.ssh/authorized_keys** file in the account on the remote system they want to access. If the user **larisa** on **turtle.mytrek.com** wants to access the **aleina** account on **rabbit.mytrek.com**, **larisa**'s public key from **/home/larisa/.ssh/id_dsa.pub** first must be placed in **aleina**'s **authorized_keys** file, **/home/aleina/.ssh/authorized_keys**. User **larisa** can send the key or have it copied over. A simple cat operation can append a key to the authorized key file. In the next example, the user adds the public key for **aleina** in the **larisa.pub** file to the authorized key file. The **larisa.pub** file is a copy of the **/home/larisa/.ssh/id_dsa.pub** file that the user received earlier.

```
cat larisa.pub >>  .ssh/authorized_keys
```

Loading Keys

If you regularly make connections to a variety of remote hosts, you can use the **ssh-agent** command to place private keys in memory where they can be accessed quickly to decrypt received transmissions. The **ssh-agent** command is intended for use at the beginning of a login session. For GNOME, you can use the openssh-askpass-gnome utility, invoked by **ssh-add**, which allows you to enter a password when you log in to GNOME. GNOME will automatically supply that password whenever you use an SSH client.

Although the **ssh-agent** command enables you to use private keys in memory, you also must specifically load your private keys into memory using the **ssh-add** command. **ssh-add** with no arguments loads your private key from your **.ssh/id_dsa** or **.ssh/id_rsa.pub** file. You are prompted for your passphrase for this private key. To remove the key from memory, use **ssh-add** with the **-d** option. If you have several private keys, you can load them all into memory. **ssh-add** with the **-l** option lists those currently loaded.

SSH Clients

SSH was originally designed to replace remote access operations, such as rlogin, rcp, and Telnet, which perform no encryption and introduce security risks such as transmitting passwords in clear text. You can also use SSH to encode X server sessions as well as FTP transmissions (sftp). The ssh-clients package contains corresponding SSH clients to replace these applications. With slogin or ssh, you can log in from a remote host to execute commands and run applications, much as you can with rlogin and rsh. With scp, you can copy files between the remote host and a network host, just as with rcp. With sftp, you can transfer FTP files secured by encryption.

ssh

With ssh, you can remotely log in from a local client to a remote system on your network operating as the SSH server. The term *local client* here refers to one outside the network, such as your home computer, and the term *remote* refers to a host system on the network to which you are connecting. In effect, you connect from your local system to the remote network host. It is designed to replace rlogin, which performs remote logins, and rsh, which executes remote commands. With ssh, you can log in from a local site to a remote host on

your network and then send commands to be executed on that host. The ssh command is also capable of supporting X Window System connections. This feature is automatically enabled if you make an ssh connection from an X Window System environment, such as GNOME or KDE. A connection is set up for you between the local X server and the remote X server. The remote host sets up a dummy X server and sends any X Window System data through it to your local system to be processed by your own local X server.

The ssh login operation function is much like the **rlogin** command. You enter the **ssh** command with the address of the remote host, followed by a **-l** option and the login name (username) of the remote account you are logging in to. The following example logs in to the **aleina** user account on the **rabbit.mytrek.com** host:

```
ssh rabbit.mytrek.com -l aleina
```

You can also use the username in an address format with ssh, as in

```
ssh aleian@rabbit.mytrek.com
```

The following listing shows how the user **george** accesses the **cecelia** account on **turtle.mytrek.com**:

```
[george@turtle george]$ ssh turtle.mytrek.com -l cecelia
cecelia@turtle.mytrek.com's password:
[cecelia@turtle cecelia]$
```

A variety of options are available to enable you to configure your connection. Most have corresponding configuration options that can be set in the configuration file. For example, with the **-c** option, you can designate which encryption method you want to use, for instance, **idea**, **des**, **blowfish**, or **arcfour**. With the **-i** option, you can select a particular private key to use. The **-C** option enables you to have transmissions compressed at specified levels (see the **ssh** Man page for a complete list of options).

scp

You use scp to copy files from one host to another on a network. Designed to replace rcp, scp uses ssh to transfer data and employs the same authentication and encryption methods. If authentication requires it, scp requests a password or passphrase. The scp program operates much like rcp. Directories and files on remote hosts are specified using the username and the host address before the filename or directory. The username specifies the remote user account that scp is accessing, and the host is the remote system where that account is located. You separate the user from the host address with an @, and you separate the host address from the file or directory name with a colon. The following example copies the file **party** from a user's current directory to the user **aleina**'s **birthday** directory, located on the **rabbit.mytrek.com** host:

```
scp party aleina@rabbit.mytrek.com:/birthday/party
```

Of particular interest is the **-r** option (recursive), which enables you to copy whole directories. See the **scp** Man page for a complete list of options. In the next example, the user copies the entire **reports** directory to the user **justin**'s **projects** directory:

```
scp -r reports justin@rabbit.mytrek.com:/projects
```

In the next example, the user **george** copies the **mydoc1** file from the user **cecelia**'s home directory:

```
[george@turtle george]$ scp cecelia@turtle.mytrek.com:mydoc1  .
cecelia@turtle.mytrek.com's password:
mydoc1     0% |                              |   0 --:--
ETA
mydoc1   100% |*****************************|  17 00:00
[george@turtle george]$
```

From a Windows system, you can also use **scp** clients such as **winscp**, which will interact with Linux scp-enabled systems.

sftp and sftp-server

With **sftp**, you can transfer FTP files secured by encryption. The **sftp** program uses the same commands as **ftp**. This client, which works only with ssh version 2, operates much like **ftp**, with many of the same commands. Use **sftp** instead of **ftp** to invoke the sftp client.

```
sftp ftp.redhat.com
```

To use the sftp client to connect to an FTP server, that server needs to be operating the sftp-server application. The ssh server invokes sftp-server to provide encrypted FTP transmissions to those using the sftp client. The sftp server and client use the SSH File Transfer Protocol (SFTP) to perform FTP operations securely.

Port Forwarding (Tunneling)

If, for some reason, you can connect to a secure host only by going through an insecure host, ssh provides a feature called port forwarding. With *port forwarding*, you can secure the insecure segment of your connection. This involves simply specifying the port at which the insecure host is to connect to the secure one. This sets up a direct connection between the local host and the remote host, through the intermediary insecure host. Encrypted data is passed through directly. This process is referred to as tunneling, creating a secure tunnel of encrypted data through connected servers.

You can set up port forwarding to a port on the remote system or to one on your local system. To forward a port on the remote system to a port on your local system, use **ssh** with the **-R** option, followed by an argument holding the local port, the remote host address, and the remote port to be forwarded, each separated from the next by a colon. This works by allocating a socket to listen to the port on the remote side. Whenever a connection is made to this port, the connection is forwarded over the secure channel, and a connection is made to a remote port from the local machine. In the following example, port 22 on the local system is connected to port 23 on the **rabbit.mytrek.com** remote system:

```
ssh -R 22:rabbit.mytrek.com:23
```

To forward a port on your local system to a port on a remote system, use the **ssh -L** option, followed by an argument holding the local port, the remote host address, and the remote port to be forwarded, each two arguments separated by a colon. A socket is allocated to listen to the port on the local side. Whenever a connection is made to this port, the connection is forwarded over the secure channel and a connection is made to the remote

port on the remote machine. In the following example, port 22 on the local system is connected to port 23 on the **rabbit.mytrek.com** remote system:

```
ssh -L 22:rabbit.mytrek.com:23
```

You can use the LocalForward and RemoteForward options in your **.ssh/config** file to set up port forwarding for particular hosts or to specify a default for all hosts you connect to.

SSH Configuration

The SSH configuration file for each user is in their **.ssh/config** file. The **/etc/ssh/ssh_config** file is used to set sitewide defaults. In the configuration file, you can set various options, as listed in the **ssh_config** Man document. The configuration file is designed to specify options for different remote hosts to which you might connect. It is organized into segments, where each segment begins with the keyword **HOST**, followed by the IP address of the host. The following lines hold the options you have set for that host. A segment ends at the next **HOST** entry. Of particular interest are the **User** and **Cipher** options. Use the **User** option to specify the names of users on the remote system who are allowed access. With the **Cipher** option, you can select which encryption method to use for a particular host. Encryption methods include IDEA, DES (standard), triple-DES (3DES), Blowfish (128 bit), Arcfour (RSA's RC4), and Twofish. The following example allows access from **larisa** at **turtle.mytrek.com** and uses Blowfish encryption for transmissions:

```
Host turtle.mytrek.com
     User larisa
     Compression no
     Cipher blowfish
```

To specify global options that apply to any host you connect to, create a **HOST** entry with the asterisk as its host, **HOST ***. This entry must be placed at the end of the configuration file because an option is changed only the first time it is set. Any subsequent entries for an option are ignored. Because a host matches on both its own entry and the global one, its specific entry should come before the global entry. The asterisk (*) and the question mark (?) are both wildcard matching operators that enable you to specify a group of hosts with the same suffix or prefix.

```
Host *
  FallBackToRsh yes
  KeepAlive no
  Cipher idea
```

Kerberos

User authentication can further be controlled for certain services by Kerberos servers, discussed the next section. Kerberos authentication provides another level of security whereby individual services can be protected, allowing use of a service only to users who are cleared for access. Kerberos servers are all enabled and configured with **authconfig-gtk** (Authentication in the System Settings menu).

Kerberos is a network authentication protocol that provides encrypted authentication to connections between a client and a server. As an authentication protocol, Kerberos requires

a client to prove its identity using encryption methods before it can access a server. Once authenticated, the client and server can conduct all communications using encryption. Whereas firewalls protect only from outside attacks, Kerberos is designed to also protect from attacks from those inside the network. Users already within a network could try to break into local servers. To prevent this, Kerberos places protection around the servers themselves, rather than an entire network or computer. A free version is available from the Massachusetts Institute of Technology at **web.mit.edu/kerberos** under the MIT Public License, which is similar to the GNU Public License. The name *Kerberos* comes from Greek mythology and is the name of the three-headed watchdog for Hades. Be sure to check the **web.mit.edu/kerberos** site for recent upgrades and detailed documentation, including FAQs, manuals, and tutorials.

TIP *The Kerberos V5 package includes its own versions of network tools such as Telnet, RCP, FTP, and RSH. These provide secure authenticated access by remote users. The tools operate in the same way as their original counterparts. The package also contains a Kerberos version of the su administrative login command, ksu.*

Kerberos Servers

The key to Kerberos is a Kerberos server through which all requests for any server services are channeled. The Kerberos server then authenticates a client, identifying the client and validating the client's right to use a particular server. The server maintains a database of authorized users. Kerberos then issues the client an encrypted ticket that the client can use to gain access to the server. For example, if a user needs to check their mail, a request for use of the mail server is sent to the Kerberos server, which then authenticates the user and issues a ticket that is then used to access the mail server. Without a Kerberos-issued ticket, no one can access any of the servers. Originally, this process required that users undergo a separate authentication procedure for each server they wanted to access. However, users now only need to perform an initial authentication that is valid for all servers.

This process involves the use of two servers, an authentication server (AS) and a ticket-granting server (TGS). Together they make up what is known as the key distribution center (KDC). In effect, they distribute keys used to unlock access to services. The authentication server first validates a user's identity. The AS issues a ticket called the ticket-granting ticket (TGT) that allows the user to access the ticket-granting server. The TGS then issues the user another ticket to actually access a service. This way, the user never has any direct access of any kind to a server during the authentication process. The process is somewhat more complex than described. An authenticator using information such as the current time, a checksum, and an optional encryption key is sent along with the ticket and is decrypted with the session key. This authenticator is used by a service to verify your identity.

NOTE *You can view your list of current tickets with the* **`klist`** *command.*

Authentication Process

The authentication server validates a user using information in its user database. Each user needs to be registered in the authentication server's database. The database will include a user password and other user information. To access the authentication server, the user

provides the username and the password. The password is used to generate a user key with which communication between the AS and the user is encrypted. The user will have their own copy of the user key with which to decrypt communications. The authentication process is illustrated in Figure 19-2.

Accessing a service with Kerberos involves the following steps:

1. The user has to be validated by the authentication server and granted access to the ticket-granting server with a ticket access key. You do this by issuing the **`kinit`** command, which will ask you to enter your Kerberos username and then send it on to the authentication server (the Kerberos username is usually the same as your username).

   ```
   kinit
   ```

2. The AS generates a ticket-granting ticket with which to access the ticket-granting server. This ticket will include a session key that will be used to let you access the TGS. The TGT is sent back to you encrypted with your user key (password).
3. The **`kinit`** program then prompts you to enter your Kerberos password, which it then uses to decrypt the TGT. You can manage your Kerberos password with the **`kpasswd`** command.
4. Now you can use a client program such as a mail client program to access the mail server, for instance. When you do so, the TGT accesses the TGS, which then generates a ticket for accessing the mail server. The TGS generates a new session key for use with just the mail server. This is provided in the ticket sent to you for accessing the mail server. In effect, there is a TGT session key used for accessing the TGS, and a mail session key used for accessing the mail server. The ticket for the mail server is sent to you encrypted with the TGS session key.

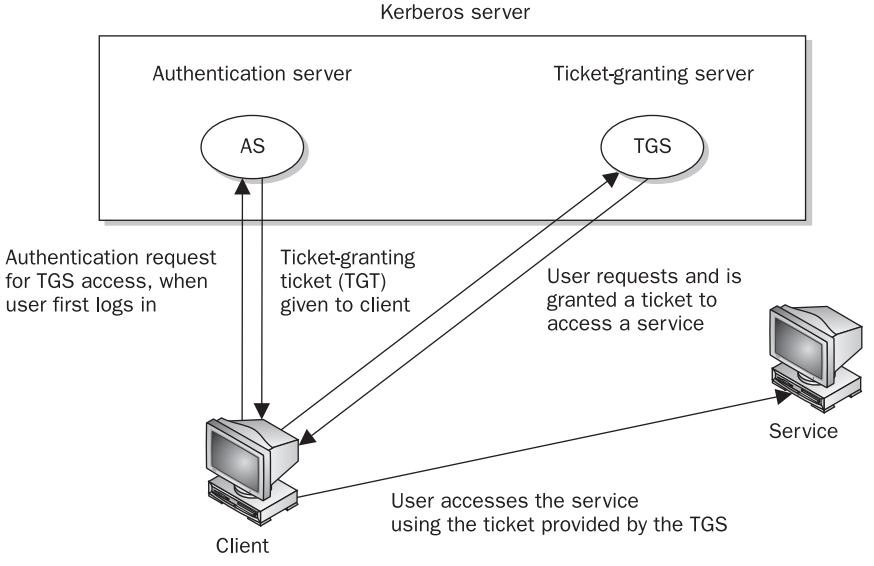


Figure 19-2 Kerberos authentication

5. The client then uses the mail ticket received from the TGS to access the mail server.
6. If you want to use another service such as FTP, when your FTP client sends a request to the TGS for a ticket, the TGS will automatically obtain authorization from the authentication server and issue an FTP ticket with an FTP session key. This kind of support remains in effect for a limited period of time, usually several hours, after which you again have to use `kinit` to undergo the authentication process and access the TGS. You can manually destroy any tickets you have with the **kdestroy** command.

NOTE *With Kerberos v5 (version 5), a Kerberos login utility is provided whereby users are automatically granted ticket-granting tickets when they log in normally. This avoids the need to use kinit to manually obtain a TGT.*

Kerberized Services

Setting up a particular service to use Kerberos (known as Kerberizing) can be a complicated process. A Kerberized service needs to check the user's identity and credentials, check for a ticket for the service, and if one is not present, obtain one. Once they are set up, use of Kerberized services is nearly transparent to the user. Tickets are automatically issued and authentication carried out without any extra effort by the user. The **/etc/services** file should contain a listing of specific Kerberized services. These are services such as **kpasswd**, **ksu**, and **klogin** that provide Kerberos password, superuser access, and login services.

Kerberos also provides its own kerberized network tools for ftp, rsh, rcp, and rlogin. These are located at **/usr/kerberos/bin**, and most have the same name as the original network tools. Priority is given to Kerberos tools by the **/etc/profile.d/krb5-workstation.sh** script, which places the Kerberos directory before all others in the user's PATH variable.

Configuring Kerberos Servers

Installing and configuring a Kerberos server is also a complex process. Carefully check the documentation for installing the current versions. Some of the key areas are listed here. In the Kerberos configuration file, **krb5.conf**, you can set such features as the encryption method used and the database name. When installing Kerberos, be sure to carefully follow the instructions for providing administrative access. To run Kerberos, you start the Kerberos server with the **service** command and the **krb5kdc**, **kadmin**, and **krb524** scripts.

You will need to configure the server for your network, along with clients for each host (the **krb5-server** package for servers and **krb5-workstation** for clients). To configure your server, you first specify your Kerberos realm and domain by manually replacing the lowercase **example.com** and the uppercase **EXAMPLE.COM** entries in the **/etc/krb5.conf** and **/var/kerberos/krb5kdc/kdc.conf** files with your own domain name. Maintain the same case for each entry. Realms are specified in uppercase, and simple host and domain names are in lowercase. You then create a database with the **kdb5_util** command and the **create** option. You will be prompted to enter a master key.

```
kdb5_util create -s
```

Full administrative access to the server is controlled by user entries in the **var/kerberos/krb5kdc/kadm5.acl** file. Replace the **EXAMPLE.COM** text with your Kerberos realm (usually

your domain name in uppercase). You then need to add a local principal, a local user with full administrative access from the host the server runs on. Start the kadmin.local tool and use the **addprinc** command to add the local principal. You can then start your **krb5kdc**, **kadmin**, and **krb524** servers.

On each client host, use the kadmin tool with the **addprincipal** command to add a principal for the host. Also add a host principal for each host on your network with the **host/** qualifier, as in **host/rabbit.mytrek.com**. You can use the **-randkey** option to specify a random key. Then save local copies of the host keys, using the **ktadd** command to save them in its **/etc/krb5.keytab** file. Each host needs to also have the same **/etc/krb5.conf** configuration file on its system, specifying the Kerberos server and the kdc host.

NOTE *When you configure Kerberos with the Authentication tool, you will be able to enter the realm, kdc server, and Kerberos server. Default entries will be displayed using the domain "example.com." Be sure to specify the realm in uppercase. A new entry for your realm will be made in the realms segment of the* ***/etc/krb5.conf,*** *listing the kdc and server entries you made.*

CHAPTER 20

Firewalls

Most systems currently connected to the Internet are open to attempts by outside users to gain unauthorized access. Outside users can try to gain access directly by setting up an illegal connection, by intercepting valid communications from users remotely connected to the system, or by pretending to be a valid user. Firewalls, encryption, and authentication procedures are ways of protecting against such attacks. A *firewall* prevents any direct unauthorized attempts at access, *encryption* protects transmissions originating from authorized remote users, and *authentication* verifies that a user requesting access has the right to do so. The current Linux kernel incorporates support for firewalls using the Netfilter (IPtables) packet filtering package (the previous version, IP Chains, is used on older kernel versions). To implement a firewall, you simply provide a series of rules to govern what kind of access you want to allow on your system. If that system is also a gateway for a private network, the system's firewall capability can effectively help protect the network from outside attacks.

To provide protection for remote communications, transmission can be simply encrypted. For Linux systems, you can use the Secure Shell (SSH) suite of programs to encrypt any transmissions, preventing them from being read by anyone else. Kerberos authentication provides another level of security whereby individual services can be protected, allowing use of a service only to users who are cleared for access. Outside users may also try to gain unauthorized access through any Internet services you may be hosting, such as a website. In such a case, you can set up a proxy to protect your site from attack. For Linux systems, use Squid proxy software to set up a proxy to protect your web server. Table 20-1 lists several network security applications commonly used on Linux.

Firewalls: IPtables, NAT, and ip6tables

A good foundation for your network's security is to set up a Linux system to operate as a firewall for your network, protecting it from unauthorized access. You can use a firewall to implement either packet filtering or proxies. *Packet filtering* is simply the process of deciding whether a packet received by the firewall host should be passed on into the local network. The packet-filtering software checks the source and destination addresses of the packet and sends the packet on, if it's allowed. Even if your system is not part of a network but connects directly to the Internet, you can still use the firewall feature to control access to your system. Of course, this also provides you with much more security.

Website	Security Application
netfilter.org	Netfilter project, IPtables, and NAT
netfilter.org/ipchains	IP Chains firewall
openssh.org	Secure Shell encryption
squid-cache.org	Squid web proxy server
web.mit.edu/Kerberos	Kerberos network authentication

TABLE 20-1 Network Security Applications

With proxies, you can control access to specific services, such as web or FTP servers. You need a proxy for each service you want to control. The web server has its own web proxy, while an FTP server has an FTP proxy. Proxies can also be used to cache commonly used data, such as web pages, so that users needn't constantly access the originating site. The proxy software commonly used on Linux systems is Squid.

An additional task performed by firewalls is network address translation (NAT). Network address translation redirects packets to appropriate destinations. It performs tasks such as redirecting packets to certain hosts, forwarding packets to other networks, and changing the host source of packets to implement IP masquerading.

NOTE *The IP Chains package is the precursor to IPtables that was used on Linux systems running the 2.2 kernel. It is still in use on many Linux systems. The Linux website for IP Chains, which is the successor to ipfwadm used on older versions of Linux, is currently netfilter.org/ipchains. IP Chains is no longer included with many Linux distributions.*

The Netfilter software package implements both packet filtering and NAT tasks for the Linux 2.4 kernel and above. The Netfilter software is developed by the Netfilter Project, which you can find out more about at **netfilter.org**.

IPtables

The command used to execute packet filtering and NAT tasks is **`iptables`**, and the software is commonly referred to as simply IPtables. However, Netfilter implements packet filtering and NAT tasks separately using different tables and commands. A table will hold the set of commands for its application. This approach streamlines the packet-filtering task, letting IPtables perform packet-filtering checks without the overhead of also having to do address translations. NAT operations are also freed from being mixed in with packet-filtering checks. You use the **`iptables`** command for both packet filtering and NAT tasks, but for NAT you add the **`-nat`** option. The IPtables software can be built directly into the kernel or loaded as a kernel module, **iptable_filter.o**.

ip6tables

The ip6tables package provides support for IPv6 addressing. It is identical to IPtables except that it allows the use of IPv6 addresses instead of IPv4 addresses. Both filter and mangle tables are supported in ip6tables, but not NAT tables. The filter tables support the same options and commands as in IPtables. The mangle tables will allow specialized packet

changes like those for IPtables, using PREROUTING, INPUT, OUTPUT, FORWARD, and POSTROUTING rules. Some extensions have ipv6 labels for their names, such as ipv6-icmp, which corresponds to the IPtables **icmp** extension. The **ipv6headers** extension is used to select IPv6 headers.

Modules

Unlike its predecessor, IP Chains, Netfilter is designed to be modularized and extensible. Capabilities can be added in the form of modules such as the state module, which adds connection tracking. Most modules are loaded as part of the IPtables service. Others are optional; you can elect to load them before installing rules. The IPtables modules are located at **/usr/lib/*kernel-version*/kernel/net/ipv4/netfilter**, where *kernel-version* is your kernel number. For IPv6 modules, check the **ipv6/netfilter** directory. Modules that load automatically will have an **ipt_** prefix, and optional ones have just an **ip_** prefix. If you are writing your own IPtables script, you would have to add **`modpobe`** commands to load optional modules directly.

Packet Filtering

Netfilter is essentially a framework for packet management that can check packets for particular network protocols and notify parts of the kernel listening for them. Built on the Netfilter framework is the packet selection system implemented by IPtables. With IPtables, different tables of rules can be set up to select packets according to differing criteria. Netfilter currently supports three tables: filter, nat, and mangle. Packet filtering is implemented using a filter table that holds rules for dropping or accepting packets. Network address translation operations such as IP masquerading are implemented using the NAT table that holds IP masquerading rules. The mangle table is used for specialized packet changes. Changes can be made to packets before they are sent out, when they are received, or as they are being forwarded. This structure is extensible in that new modules can define their own tables with their own rules. It also greatly improves efficiency. Instead of all packets checking one large table, they access only the table of rules they need to.

IP table rules are managed using the **`iptables`** command. For this command, you will need to specify the table you want to manage. The default is the filter table, which need not be specified. You can list the rules you have added at any time with the **`-L`** and **`-n`** options, as shown here. The **`-n`** option says to use only numeric output for both IP addresses and ports, avoiding a DNS lookup for hostnames. You could, however, just use the **`-L`** option to see the port labels and hostnames:

```
iptables -L -n
```

NOTE *In IPtables commands, chain names have to be entered in uppercase, as with the chain names INPUT, OUTPUT, and FORWARD.*

Chains

Rules are combined into different chains. The kernel uses chains to manage packets it receives and sends out. A *chain* is simply a checklist of rules. These rules specify what action to take for packets containing certain headers. The rules operate with an if-then-else structure.

Target	Function
ACCEPT	Allow packet to pass through the firewall.
DROP	Deny access by the packet.
REJECT	Deny access and notify the sender.
QUEUE	Send packets to user space.
RETURN	Jump to the end of the chain and let the default target process it.

TABLE 20-2 IPtables Targets

If a packet does not match the first rule, the next rule is then checked, and so on. If the packet does not match any rules, the kernel consults chain policy. Usually, at this point the packet is rejected. If the packet does match a rule, it is passed to its target, which determines what to do with the packet. The standard targets are listed in Table 20-2. If a packet does not match any of the rules, it is passed to the chain's default target.

Targets

A *target* can, in turn, be another chain of rules, even a chain of user-defined rules. A packet could be passed through several chains before finally reaching a target. In the case of user-defined chains, the default target is always the next rule in the chains from which it was called. This sets up a procedure- or function call–like flow of control found in programming languages. When a rule has a user-defined chain as its target, when activated, that user-defined chain is executed. If no rules are matched, execution returns to the next rule in the originating chain.

TIP *Specialized targets and options can be added by means of kernel patches provided by the Netfilter site. For example, the SAME patch returns the same address for all connections. A patch-o-matic option for the Netfilter make file will patch your kernel source code, adding support for the new target and options. You can then rebuild and install your kernel.*

Firewall and NAT Chains

The kernel uses three firewall chains: INPUT, OUTPUT, and FORWARD. When a packet is received through an interface, the INPUT chain is used to determine what to do with it. The kernel then uses its routing information to decide where to send it. If the kernel sends the packet to another host, the FORWARD chain is checked. Before the packet is actually sent, the OUTPUT chain is also checked. In addition, two NAT table chains, POSTROUTING and PREROUTING, are implemented to handle masquerading and packet address modifications. The built-in Netfilter chains are listed in Table 20-3.

Adding and Changing Rules

You add and modify chain rules using the **`iptables`** commands. An **`iptables`** command consists of the command **`iptables`**, followed by an argument denoting the command to execute. For example, **`iptables -A`** is the command to add a new rule, whereas **`iptables -D`** is the command to delete a rule. The **`iptables`** commands are listed in Table 20-4.

Chain	Description
INPUT	Rules for incoming packets
OUTPUT	Rules for outgoing packets
FORWARD	Rules for forwarded packets
PREROUTING	Rules for redirecting or modifying incoming packets, NAT table only
POSTROUTING	Rules for redirecting or modifying outgoing packets, NAT table only

TABLE 20-3 Netfilter Built-in Chains

The following command simply lists the chains along with their rules currently defined for your system. The output shows the default values created by **iptables** commands.

```
iptables -L -n
Chain input (policy ACCEPT):
Chain forward (policy ACCEPT):
Chain output (policy ACCEPT):
```

To add a new rule to a chain, you use **-A**. Use **-D** to remove it, and **-R** to replace it. Following the command, list the chain to which the rule applies, such as the INPUT, OUTPUT, or FORWARD chain, or a user-defined chain. Next, you list different options that specify the actions you want taken (most are the same as those used for IP Chains, with a few exceptions).

Option	Function
-A *chain*	Appends a rule to a chain.
-D *chain* [*rulenum*]	Deletes matching rules from a chain. Deletes rule *rulenum* (1 = first) from *chain*.
-I *chain* [*rulenum*]	Inserts in *chain* as *rulenum* (default 1 = first).
-R *chain rulenum*	Replaces rule *rulenum* (1 = first) in *chain.*
-L [*chain*]	Lists the rules in *chain* or all chains.
-E [*chain*]	Renames a chain.
-F [*chain*]	Deletes (flushes) all rules in *chain* or all chains.
-R *chain*	Replaces a rule; rules are numbered from 1.
-Z [*chain*]	Zero counters in *chain* or all chains.
-N *chain*	Creates a new user-defined chain.
-X *chain*	Deletes a user-defined chain.
-P *chain target*	Changes policy on *chain* to *target.*

TABLE 20-4 IPtables Commands

The **-s** option specifies the source address attached to the packet, **-d** specifies the destination address, and the **-j** option specifies the target of the rule. The ACCEPT target will allow a packet to pass. The **-i** option now indicates the input device and can be used only with the INPUT and FORWARD chains. The **-o** option indicates the output device and can be used only for OUTPUT and FORWARD chains. Table 20-5 lists several basic options.

Option	Function
-p [!] *proto*	Specifies a protocol, such as TCP, UDP, ICMP, or ALL.
-s [!] *address*[**/***mask*] **[!]** [*port*[**:***port*]]	Specifies source address to match. With the *port* argument, you can specify the port.
--sport [!] [*port*[**:***port*]]	Specifies source port. You can specify a range of ports using the colon, *port*:*port.*
-d [!] *address*[**/***mask*] **[!]** [*port*[**:***port*]]	Specifies destination address to match. With the *port* argument, you can specify the port.
--dport [!] [*port*[**:***port*]]	Specifies destination port.
--icmp-type [!] *typename*	Specifies ICMP type.
-i [!] *name*[**+**]	Specifies an input network interface using its name (for example, **eth0**). The **+** symbol functions as a wildcard. The **+** attached to the end of the name matches all interfaces with that prefix (**eth+** matches all Ethernet interfaces). Can be used only with the INPUT chain.
-j *target* [*port*]	Specifies the target for a rule (specify [*port*] for REDIRECT target).
--to-source <*ipaddr*> [**-**<*ipaddr*>] [**:** *port*- *port*]	Used with the SNAT target, rewrites packets with new source IP address.
--to-destination <*ipaddr*> [**-**<*ipaddr*>] [**:** *port*- *port*]	Used with the DNAT target, rewrites packets with new destination IP address.
-n	Specifies numeric output of addresses and ports, used with **-L**.
-o [!] *name*[**+**]	Specifies an output network interface using its name (for example, **eth0**). Can be used only with FORWARD and OUTPUT chains.
-t *table*	Specifies a table to use, as in **-t nat** for the NAT table.
-v	Verbose mode, shows rule details, used with **-L**.
-x	Expands numbers (displays exact values), used with **-L**.
[!] -f	Matches second through last fragments of a fragmented packet.
[!] -V	Prints package version.
!	Negates an option or address.

TABLE 20-5 IPtables Options

Option	Function
`-m`	Specifies a module to use, such as state.
`--state`	Specifies options for the state module such as NEW, INVALID, RELATED, and ESTABLISHED. Used to detect packet's state. NEW references SYN packets (new connections).
`--syn`	SYN packets, new connections.
`--tcp-flags`	TCP flags: SYN, ACK, FIN, RST, URG, PS, and ALL for all flags.
`--limit`	Option for the limit module (`-m limit`). Used to control the rate of matches, matching a given number of times per second.
`--limit-burst`	Option for the limit module (`-m limit`). Specifies maximum burst before the limit kicks in. Used to control denial-of-service attacks.

TABLE 20-5 IPtables Options (*continued*)

IPtables Options

The IPtables package is designed to be extensible, and there are number of options with selection criteria that can be included with IPtables. For example, the TCP extension includes the `--syn` option that checks for SYN packets. The ICMP extension provides the `--icmp-type` option for specifying ICMP packets as those used in ping operations. The limit extension includes the `--limit` option, with which you can limit the maximum number of matching packets in a specified time period, such as a second.

In the following example, the user adds a rule to the INPUT chain to accept all packets originating from the address 192.168.0.55. Any packets that are received (`INPUT`) whose source address (`-s`) matches 192.168.0.55 are accepted and passed through (`-j ACCEPT`):

```
iptables -A INPUT -s 192.168.0.55 -j ACCEPT
```

Accepting and Denying Packets: DROP and ACCEPT

There are two built-in targets, DROP and ACCEPT. Other targets can be either user-defined chains or extensions added on, such as REJECT. Two special targets are used to manage chains, RETURN and QUEUE. RETURN indicates the end of a chain and returns to the chain it started from. QUEUE is used to send packets to user space.

```
iptables -A INPUT -s www.myjunk.com -j DROP
```

You can turn a rule into its inverse with an `!` symbol. For example, to accept all incoming packets except those from a specific address, place an `!` symbol before the `-s` option and that address. The following example will accept all packets except those from the IP address 192.168.0.45:

```
iptables -A INPUT -j ACCEPT ! -s 192.168.0.45
```

You can specify an individual address using its domain name or its IP number. For a range of addresses, you can use the IP number of their network and the network IP mask. The IP mask can be an IP number or simply the number of bits making up the mask. For example, all of the addresses in network 192.168.0 can be represented by 192.168.0.0/225.255.255.0 or by 192.168.0.0/24. To specify any address, you can use 0.0.0.0/0.0.0.0 or simply 0/0. By default, rules reference any address if no **-s** or **-d** specification exists. The following example accepts messages coming in that are from (source) any host in the 192.168.0.0 network and that are going (destination) anywhere at all (the **-d** option is left out or could be written as **-d 0/0**):

```
iptables -A INPUT -s 192.168.0.0/24  -j ACCEPT
```

The IPtables rules are usually applied to a specific network interface such as the Ethernet interface used to connect to the Internet. For a single system connected to the Internet, you will have two interfaces, one that is your Internet connection and a loopback interface (**lo**) for internal connections between users on your system. The network interface for the Internet is referenced using the device name for the interface. For example, an Ethernet card with the device name **/dev/eth0** would be referenced by the name **eth0**. A modem using PPP protocols with the device name **/dev/ppp0** would have the name **ppp0**. In IPtables rules, you use the **-i** option to indicate the input device; it can be used only with the INPUT and FORWARD chains. The **-o** option indicates the output device and can be used only for OUTPUT and FORWARD chains. Rules can then be applied to packets arriving and leaving on particular network devices. In the following examples, the first rule references the Ethernet device **eth0**, and the second, the localhost:

```
iptables -A INPUT -j DROP -i eth0 -s 192.168.0.45
iptables -A INPUT -j ACCEPT  -i lo
```

User-Defined Chains

With IPtables, the FORWARD and INPUT chains are evaluated separately. One does not feed into the other. This means that if you want to completely block certain addresses from passing through your system, you will need to add both a FORWARD rule and an INPUT rule for them.

```
iptables -A INPUT -j DROP -i eth0 -s 192.168.0.45
iptables -A FORWARD -j DROP -i eth0 -s 192.168.0.45
```

A common method for reducing repeated INPUT and FORWARD rules is to create a user chain that both the INPUT and FORWARD chains feed into. You define a user chain with the **-N** option. The next example shows the basic format for this arrangement. A new chain is created called incoming (it can be any name you choose). The rules you define for your FORWARD and INPUT chains are now defined for the incoming chain. The INPUT and FORWARD chains then use the incoming chain as a target, jumping directly to it and using its rules to process any packets they receive.

```
iptables -N incoming

iptables -A incoming -j DROP -i eth0 -s 192.168.0.45
iptables -A incoming -j ACCEPT  -i lo
```

```
iptables -A FORWARD -j incoming
iptables -A INPUT -j incoming
```

ICMP Packets

Firewalls often block certain Internet Control Message Protocol (ICMP) messages. ICMP redirect messages, in particular, can take control of your routing tasks. You need to enable some ICMP messages, however, such as those needed for ping, traceroute, and particularly destination-unreachable operations. In most cases, you always need to make sure destination-unreachable packets are allowed; otherwise, domain name queries could hang. Some of the more common ICMP packet types are listed in Table 20-6. You can enable an ICMP type of packet with the **`--icmp-type`** option, which takes as its argument a number or a name representing the message. The following examples enable the use of echo-reply, echo-request, and destination-unreachable messages, which have the numbers 0, 8, and 3:

```
iptables -A INPUT -j ACCEPT  -p icmp -i eth0 --icmp -type  echo-reply -d 10.0.0.1
iptables -A INPUT -j ACCEPT  -p icmp -i eth0 --icmp-type  echo-request -d 10.0.0.1
iptables -A INPUT -j ACCEPT -p icmp -i eth0 --icmp-type  destination-unreachable -d
10.0.0.1
```

Their rule listing will look like this:

```
ACCEPT     icmp --  0.0.0.0/0            10.0.0.1            icmp type 0
ACCEPT     icmp --  0.0.0.0/0            10.0.0.1            icmp type 8
ACCEPT     icmp --  0.0.0.0/0            10.0.0.1            icmp type 3
```

Ping operations need to be further controlled to avoid the ping-of-death security threat. You can do this several ways. One way is to deny any ping fragments. Ping packets are normally very small. You can block ping-of-death attacks by denying any ICMP packet that is a fragment. Use the **`-f`** option to indicate fragments.

```
iptables -A INPUT -p icmp -j DROP -f
```

Another way is to limit the number of matches received for ping packets. You use the limit module to control the number of matches on the ICMP ping operation. Use **`-m limit`** to use the limit module, and **`--limit`** to specify the number of allowed matches. **`1/s`** will allow one match per second.

```
iptables -A FORWARD -p icmp --icmp-type echo-request -m limit --limit 1/s -j ACCEPT
```

Number	Name	Required By
0	echo-reply	ping
3	destination-unreachable	Any TCP/UDP traffic
5	redirect	Routing, if not running routing daemon
8	echo-request	ping
11	time-exceeded	traceroute

Table 20-6 Common ICMP Packets

Controlling Port Access

If your system is hosting an Internet service, such as a web or FTP server, you can use IPtables to control access to it. You can specify a particular service by using the source port (`--sport`) or destination port (`--dport`) options with the port that the service uses. IPtables lets you use names for ports such as **www** for the web server port. The names of services and the ports they use are listed in the **/etc/services** file, which maps ports to particular services. For a domain name server, the port would be **domain**. You can also use the port number if you want, preceding the number with a colon. The following example accepts all messages to the web server located at 192.168.0.43:

```
iptables -A INPUT -d 192.168.0.43 --dport www -j ACCEPT
```

You can also use port references to protect certain services and deny others. This approach is often used if you are designing a firewall that is much more open to the Internet, letting users make freer use of Internet connections. Certain services you know can be harmful, such as Telnet and NTP, can be denied selectively. For example, to deny any kind of Telnet operation on your firewall, you can drop all packets coming in on the Telnet port, 23. To protect NFS operations, you can deny access to the port used for the portmapper, 111. You can use either the port number or the port name.

```
# deny outside access to portmapper port on firewall.
iptables -A arriving  -j DROP -p tcp -i eth0  --dport 111
# deny outside access to telnet port on firewall.
iptables -A arriving  -j DROP -p tcp -i eth0  --dport telnet
```

The rule listing will look like this:

```
DROP       tcp  --  0.0.0.0/0     0.0.0.0/0     tcp dpt:111
DROP       tcp  --  0.0.0.0/0     0.0.0.0/0     tcp dpt:23
```

One port-related security problem is access to your X server on the XFree86 ports that range from 6000 to 6009. On a relatively open firewall, these ports could be used to illegally access your system through your X server. A range of ports can be specified with a colon, as in 6000:6009. You can also use x11 for the first port, x11:6009. Sessions on the X server can be secured by using SSH, which normally accesses the X server on port 6010.

```
iptables -A arriving  -j DROP -p tcp -i eth0  --dport 6000:6009
```

Common ports checked and their labels are shown here:

Service	Port Number	Port Label
Auth	113	auth
Finger	79	finger
FTP	21	ftp
NTP	123	ntp
Portmapper	111	sunrpc
Telnet	23	telnet
Web server	80	www
XFree86	6000:6009	x11:6009

Packet States: Connection Tracking

One of the more useful extensions is the state extension, which can easily detect tracking information for a packet. Connection tracking maintains information about a connection such as its source, destination, and port. It provides an effective means for determining which packets belong to an established or related connection. To use connection tracking, you specify the state module first with **`-m state`**. Then you can use the **`--state`** option. Here you can specify any of the following states:

State	Description
NEW	A packet that creates a new connection
ESTABLISHED	A packet that belongs to an existing connection
RELATED	A packet that is related to, but not part of, an existing connection, such as an ICMP error or a packet establishing an FTP data connection
INVALID	A packet that could not be identified for some reason
RELATED+REPLY	A packet that is related to an established connection but is not part of one directly

If you are designing a firewall that is meant to protect your local network from any attempts to penetrate it from an outside network, you may want to restrict packets coming in. Simply denying access by all packets is unfeasible because users connected to outside servers—say, on the Internet—must receive information from them. You can, instead, deny access by a particular kind of packet used to initiate a connection. The idea is that an attacker must initiate a connection from the outside. The headers of these kinds of packets have their SYN bit set on and their FIN and ACK bits empty. The state module's NEW state matches on any such SYN packet. By specifying a DROP target for such packets, you deny access by any packet that is part of an attempt to make a connection with your system. Anyone trying to connect to your system from the outside is unable to do so. Users on your local system who have initiated connections with outside hosts can still communicate with them. The following example will drop any packets trying to create a new connection on the **eth0** interface, though they will be accepted on any other interface:

```
iptables -A INPUT -m state --state NEW -i eth0 -j DROP
```

You can use the **`!`** operator on the **eth0** device combined with an ACCEPT target to compose a rule that will accept any new packets except those on the **eth0** device. If the **eth0** device is the only one that connects to the Internet, this still effectively blocks outside access. At the same time, input operation for other devices such as your localhost are free to make new connections. This kind of conditional INPUT rule is used to allow access overall with exceptions. It usually assumes that a later rule such as a chain policy will drop remaining packets.

```
iptables -A INPUT -m state --state NEW ! -i eth0 -j ACCEPT
```

The next example will accept any packets that are part of an established connection or related to such a connection on the **eth0** interface:

```
iptables -A INPUT -m state --state ESTABLISHED,RELATED -j ACCEPT
```

TIP *You can use the iptstate tool to display the current state table.*

Specialized Connection Tracking: ftp, irc, Amanda, tftp

To track certain kinds of packets, IPtables uses specialized connection tracking modules. These are optional modules that you have to have loaded manually. To track passive FTP connections, you would have to load the ip_conntrack_ftp module. To add NAT table support, you would also load the ip_nat_ftp module. For IRC connections, you use ip_conntrack_irc and ip_nat_irc. There are corresponding modules for Amanda (the backup server) and TFTP (Trivial FTP).

If you are writing your own IPtables script, you would have to add **`modprobe`** commands to load the modules.

```
modprobe ip_conntrack ip_conntrack_ftp ip_nat_ftp
modprobe ip_conntrack_amanda ip_nat_amanda
```

Network Address Translation (NAT)

Network address translation (NAT) is the process whereby a system will change the destination or source of packets as they pass through the system. A packet will traverse several linked systems on a network before it reaches its final destination. Normally, they will simply pass the packet on. However, if one of these systems performs a NAT on a packet, it can change the source or destination. A packet sent to a particular destination can have its destination address changed. To make this work, the system also needs to remember such changes so that the source and destination for any reply packets are altered back to the original addresses of the packet being replied to.

NAT is often used to provide access to systems that may be connected to the Internet through only one IP address. Such is the case with networking features such as IP masquerading, support for multiple servers, and transparent proxying. With IP masquerading, NAT operations will change the destination and source of a packet moving through a firewall/gateway linking the Internet to computers on a local network. The gateway has a single IP address that the other local computers can use through NAT operations. If you have multiple servers but only one IP address, you can use NAT operations to send packets to the alternate servers. You can also use NAT operations to have your IP address reference a particular server application such as a web server (transparent proxy). NAT tables are not implemented for ip6tables.

Adding NAT Rules

Packet selection rules for NAT operations are added to the NAT table managed by the **`iptables`** command. To add rules to the NAT table, you have to specify the NAT table with the **`-t`** option. Thus to add a rule to the NAT table, you would have to specify the NAT table with the **`-t nat`** option as shown here:

```
iptables -t nat
```

With the **`-L`** option, you can list the rules you have added to the NAT table:

```
iptables -t nat -L -n
```

Adding the **-n** option will list IP addresses and ports in numeric form. This will speed up the listing, as IPtables will not attempt to do a DNS lookup to determine the hostname for the IP address.

NAT Targets and Chains

In addition, there are two types of NAT operations: source NAT, specified as SNAT target, and destination NAT, specified as DNAT target. SNAT target is used for rules that alter source addresses, and DNAT target, for those that alter destination addresses.

Three chains in the NAT table are used by the kernel for NAT operations: PREROUTING, POSTROUTING, and OUTPUT. PREROUTING is used for destination NAT (DNAT) rules, which are packets that are arriving. POSTROUTING is used for source NAT (SNAT) rules, which are for packets leaving. OUTPUT is used for destination NAT rules for locally generated packets.

As with packet filtering, you can specify source (**-s**) and destination (**-d**) addresses, as well as the input (**-i**) and output (**-o**) devices. The **-j** option will specify a target such as MASQUERADE. You implement IP masquerading by adding a MASQUERADE rule to the POSTROUTING chain:

```
# iptables -t nat -A POSTROUTING -o eth0 -j MASQUERADE
```

To change the source address of a packet leaving your system, you use the POSTROUTING rule with the SNAT target. For the SNAT target, you use the **--to-source** option to specify the source address:

```
# iptables -t nat -A POSTROUTING -o eth0 -j SNAT --to-source 192.168.0.4
```

To change the destination address of packets arriving on your system, you use the PREROUTING rule with the DNAT target and the **--to-destination** option:

```
# iptables -t nat -A PRETROUTING -i eth0 \
             -j DNAT --to-destination 192.168.0.3
```

Specifying a port lets you change destinations for packets arriving on a particular port. In effect, this lets you implement port forwarding. In the next example, every packet arriving on port 80 (the web service port) is redirected to 10.0.0.3, which in this case would be a system running a web server.

```
# iptables -t nat -A PRETROUTING -i eth0 -dport 80 \
             -j DNAT --to-destination 10.0.0.3
```

With the TOS and MARK targets, you can mangle the packet to control its routing or priority. A TOS target sets the type of service for a packet, which can set the priority using criteria such as normal-service, minimize-cost, or maximize-throughput, among others.

The targets valid only for the NAT table are shown here:

SNAT	Modify source address, use **`--to-source`** option to specify new source address.
DNAT	Modify destination address, use **`--to-destination`** option to specify new destination address.
REDIRECT	Redirect a packet.
MASQUERADE	IP masquerading.
MIRROR	Reverse source and destination and send back to sender.
MARK	Modify the Mark field to control message routing.

NAT Redirection: Transparent Proxies

NAT tables can be used to implement any kind of packet redirection, a process transparent to the user. Redirection is commonly used to implement a transparent proxy. Redirection of packets is carried out with the REDIRECT target. With transparent proxies, packets received can be automatically redirected to a proxy server. For example, packets arriving on the web service port, 80, can be redirected to the Squid proxy service port, usually 3128. This involves a command to redirect a packet, using the REDIRECT target on the PREROUTING chain:

```
iptables -t nat -A PREROUTING -i eth1 --dport 80 -j REDIRECT --to-port 3128
```

Packet Mangling: The Mangle Table

The *packet mangling* table is used to actually modify packet information. Rules applied specifically to this table are often designed to control the mundane behavior of packets, like routing, connection size, and priority. Rules that actually modify a packet, rather than simply redirecting or stopping it, can be used only in the mangle table. For example, the TOS target can be used directly in the mangle table to change the Type of Service field to modify a packet's priority. A TCPMSS target can be set to modify the allowed size of packets for a connection. The ECN target lets you work around ECN black holes, and the DSCP target will let you change DSCP bits. Several extensions such as the ROUTE extension will change a packet, in this case, rewriting its destination rather than just redirecting it.

The mangle table is indicated with the **`-t mangle`** option. Use the following command to see what chains are listed in your mangle table:

```
iptables -t mangle  -L
```

Several mangle table targets are shown here:

TOS	Modify the Type of Service field to manage the priority of the packet.
TCPMSS	Modify the allowed size of packets for a connection, enabling larger transmissions.
ECN	Remove ECN black hole information.
DSCP	Change DSCP bits.
ROUTE	Extension TARGET to modify destination information in the packet.

NOTE *The IPtables package is designed to be extensible, allowing customized targets to be added easily. This involves applying patches to the kernel and rebuilding it. See* ***netfilter.org*** *for more details, along with a listing of extended targets.*

IPtables Scripts

Though you can enter IPtables rules from the shell command line, when you shut down your system, these commands will be lost. You will most likely need to place your IPtables rules in a script that can then be executed directly. This way you can edit and manage a complex set of rules, adding comments and maintaining their ordering.

An IPtables Script Example: IPv4

You now have enough information to create a simple IPtables script that will provide basic protection for a single system connected to the Internet. The following script, `myfilter`, provides an IPtables filtering process to protect a local network and a website from outside attacks. This example uses IPtables and IPv4 addressing. For IPv6 addressing you would use ip6tables, which has corresponding commands, except for the NAT rules, which would be implemented as mangle rules.

The script configures a simple firewall for a private network (check the IPtables HOWTO for a more complex example). If you have a local network, you can adapt this script to it. In this configuration, all remote access initiated from the outside is blocked, but two-way communication is allowed for connections that users in the network make with outside systems. In this example, the firewall system functions as a gateway for a private network whose network address is 192.168.0.0 (see Figure 20-1). The Internet address is, for the sake

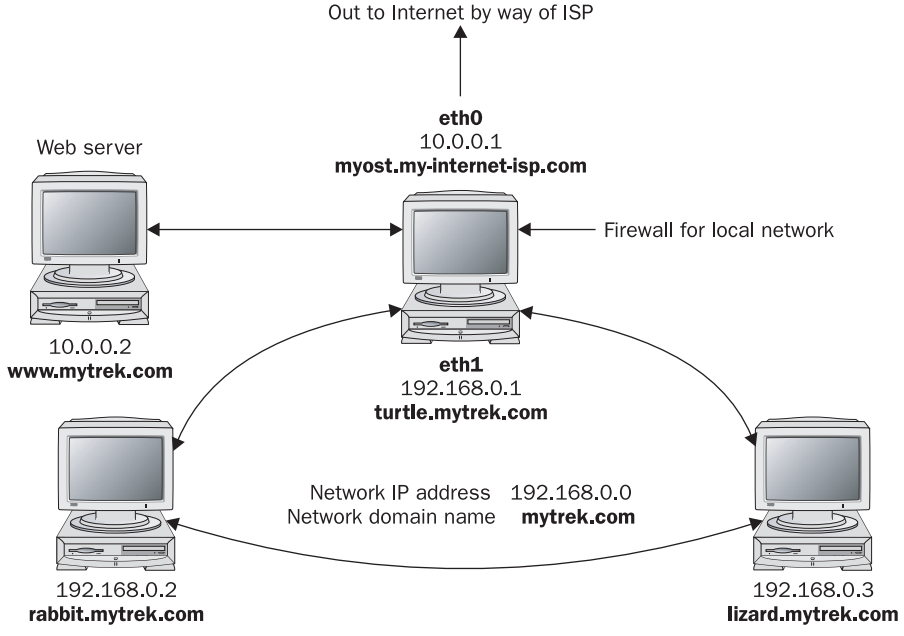


FIGURE 20-1 A network with a firewall

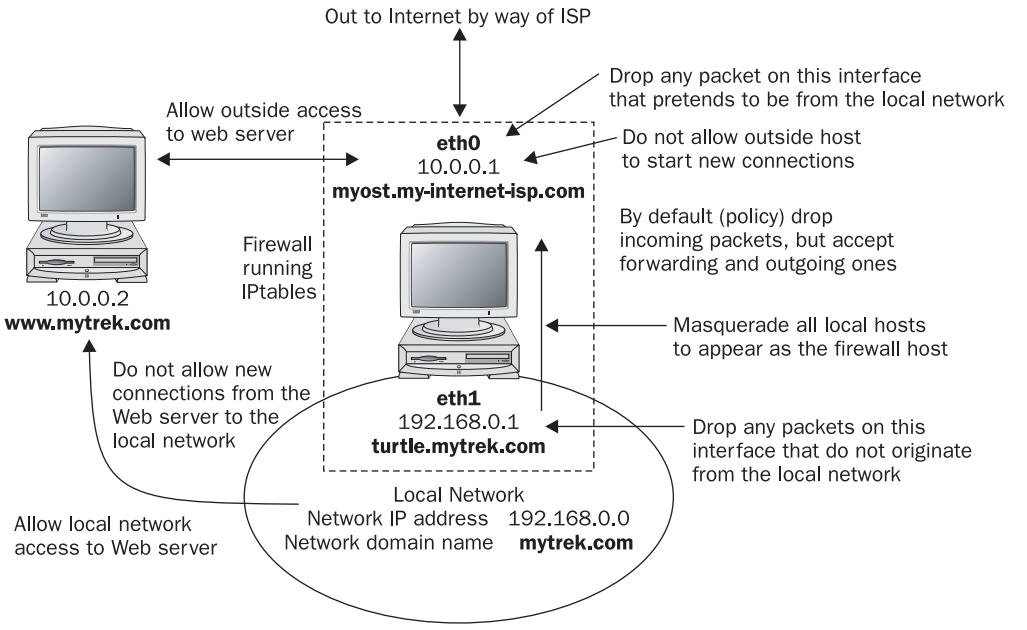


Figure 20-2 Firewall rules applied to a local network example

of this example, 10.0.0.1. The system has two Ethernet devices: one for the private network (**eth1**) and one for the Internet (**eth0**). The gateway firewall system also supports a web server at address 10.0.0.2. Entries in this example that are too large to fit on one line are continued on a second line, with the newline quoted with a backslash.

The basic rules as they apply to different parts of the network are illustrated in Figure 20-2.

myfilter

```
# Firewall Gateway system IP address is 10.0.0.1 using Ethernet device eth0
# Private network address is 192.168.0.0 using Ethernet device eth1
# Website address is 10.0.0.2
# turn off IP forwarding
echo 0 > /proc/sys/net/ipv4/ip_forward
# Flush chain rules
iptables -F INPUT
iptables -F OUTPUT
iptables -F FORWARD
# set default (policy) rules
iptables -P INPUT DROP
iptables -P OUTPUT ACCEPT
iptables -P FORWARD ACCEPT

# IP spoofing, deny any packets on the internal network that have an external
source address.
iptables -A INPUT -j LOG  -i eth1 \! -s 192.168.0.0/24
iptables -A INPUT -j DROP  -i eth1 \! -s 192.168.0.0/24
iptables -A FORWARD -j DROP  -i eth1 \! -s 192.168.0.0/24
```

```
# IP spoofing, deny any outside packets (any not on eth1) that have the
source address of the internal network
iptables -A INPUT -j DROP \! -i eth1 -s 192.168.0.0/24
iptables -A FORWARD -j DROP \! -i eth1 -s 192.168.0.0/24
# IP spoofing, deny any outside packets with localhost address
# (packets not on the lo interface (any on eth0 or eth1) that have the source
address of localhost)
iptables -A INPUT -j DROP  -i \! lo  -s  127.0.0.0/255.0.0.0
iptables -A FORWARD -j DROP  -i \! lo  -s  127.0.0.0/255.0.0.0

# allow all incoming messages for users on your firewall system
iptables -A INPUT -j ACCEPT  -i lo

# allow  communication to the web server (address 10.0.0.2), port www
iptables -A INPUT  -j ACCEPT -p tcp -i eth0  --dport www -s 10.0.0.2
# Allow  established connections from web servers to internal network
iptables -A INPUT -m state --state ESTABLISHED,RELATED -i eth0 -p tcp
--sport www -s 10.0.0.2 -d 192.168.0.0/24  -j ACCEPT
# Prevent new  connections from web servers to internal network
iptables -A OUTPUT -m state --state  NEW -o eth0 -p tcp  --sport
www -d 192.168.0.0/24  -j DROP

# allow established and related outside communication to your system
# allow outside communication to the firewall, except for ICMP packets
iptables -A INPUT -m state --state ESTABLISHED,RELATED -i eth0 -p \! icmp -j
ACCEPT
# prevent outside-initiated connections
iptables -A INPUT -m state --state NEW -i eth0 -j DROP
iptables -A FORWARD -m state --state NEW -i eth0 -j DROP
# allow all local communication to and from the firewall on eth1  from the
local network
iptables -A INPUT -j ACCEPT -p all -i eth1 -s 192.168.0.0/24
# Set up masquerading to allow internal machines access to outside network
iptables -t nat -A POSTROUTING -o eth0 -j MASQUERADE
# Accept ICMP Ping and Destination-unreachable messages
# Others will be rejected by INPUT and OUTPUT DROP policy
iptables -A INPUT -j ACCEPT  -p icmp -i eth0 --icmp-type  echo-reply
-d 10.0.0.1
iptables -A INPUT -j ACCEPT  -p icmp -i eth0 --icmp-type  echo-request
-d 10.0.0.1
iptables -A INPUT -j ACCEPT -p icmp -i eth0 --icmp-type  destination
-unreachable -d 10.0.0.1
# Turn on IP Forwarding
echo 1 > /proc/sys/net/ipv4/ip_forward
```

Initially, in the script you clear your current IPtables with the flush option (`-F`), and then set the policies (default targets) for the non–user-defined rules. IP forwarding should also be turned off while the chain rules are being set:

```
echo 0 > /proc/sys/net/ipv4/ip_forward
```

Drop Policy

First, a DROP policy is set up for the INPUT and FORWARD built-in IP chains. This means that if a packet does not meet a criterion in any of the rules to let it pass, it will be dropped.

Then both IP spoofing attacks and any attempts from the outside to initiate connections (SYN packets) are rejected. Outside connection attempts are also logged. This is a very basic configuration that can easily be refined to your own needs by adding IPtables rules.

```
iptables -P INPUT DROP
iptables -P OUTPUT ACCEPT
iptables -P FORWARD ACCEPT
```

IP Spoofing

One way to protect the private network from the IP spoofing of any packets is to check for any outside addresses on the Ethernet device dedicated to the private network. In this example, any packet on device **eth1** (dedicated to the private network) whose source address is not that of the private network (`! -s 192.168.0.0`) is denied. Also, check to see if any packets coming from the outside are designating the private network as their source. In this example, any packets with the source address of the private network on any Ethernet device other than for the private network (**eth1**) are denied. The same strategy can be applied to the localhost.

```
# IP spoofing, deny any packets on the internal network
# that have an external source address.
iptables -A INPUT -j LOG  -i eth1 \! -s 192.168.0.0/24
iptables -A INPUT -j DROP  -i eth1 \! -s 192.168.0.0/24
iptables -A FORWARD -j DROP  -i eth1 \! -s 192.168.0.0/24
# IP spoofing, deny any outside packets (any not on eth1)
# that have the source address of the internal network
iptables -A INPUT -j DROP \! -i eth1 -s 192.168.0.0/24
iptables -A FORWARD -j DROP \! -i eth1 -s 192.168.0.0/24
# IP spoofing, deny any outside packets with localhost address
# (packets not on the lo interface (any on eth0 or eth1)
# that have the source address of localhost)
iptables -A INPUT -j DROP  -i \! lo  -s  127.0.0.0/255.0.0.0
iptables -A FORWARD -j DROP  -i \! lo  -s  127.0.0.0/255.0.0.0
```

Then, you set up rules to allow all packets sent and received within your system (localhost) to pass.

```
iptables -A INPUT -j ACCEPT  -i lo
```

Server Access

For the web server, you want to allow access by outside users but block access by anyone attempting to initiate a connection from the web server into the private network. In the next example, all messages are accepted to the web server, but the web server cannot initiate contact with the private network. This prevents anyone from breaking into the local network through the web server, which is open to outside access. Established connections are allowed, permitting the private network to use the web server.

```
# allow  communication to the web server (address 10.0.0.2), port www
iptables -A INPUT  -j ACCEPT -p tcp -i eth0  --dport www -s 10.0.0.2
# Allow  established connections from web servers to internal network
iptables -A INPUT -m state --state ESTABLISHED,RELATED -i eth0 \
   -p tcp  --sport www -s 10.0.0.2 -d 192.168.0.0/24  -j ACCEPT
```

```
# Prevent new  connections from web servers to internal network
iptables -A OUTPUT -m state --state  NEW -o eth0 -p tcp \
  --sport www -d 192.168.0.1.0/24  -j DROP
```

Firewall Outside Access

To allow access by the firewall to outside networks, you allow input by all packets except for ICMP packets. These are handled later. The firewall is specified by the firewall device, **eth0**. First, your firewall should allow established and related connections to proceed, as shown here. Then you block outside access as described later.

```
# allow outside communication to the firewall,
# except for ICMP packets
iptables -A INPUT -m state --state ESTABLISHED,RELATED \
         -i eth0 -p \! icmp -j ACCEPT
```

Blocking Outside-Initiated Access

To prevent outsiders from initiating any access to your system, create a rule to block access by SYN packets from the outside using the **`state`** option with NEW. Drop any new connections on the **eth0** connection (assuming only **eth0** is connected to the Internet or outside network).

```
# prevent outside-initiated connections
iptables -A INPUT -m state --state NEW -i eth0 -j DROP
iptables -A FORWARD -m state --state NEW -i eth0 -j DROP
```

Local Network Access

To allow interaction by the internal network with the firewall, you allow input by all packets on the internal Ethernet connection, **eth1**. The valid internal network addresses are designated as the input source.

```
iptables -A INPUT -j ACCEPT -p all -i eth1 -s 192.168.0.0/24
```

Masquerading Local Networks

To implement masquerading, where systems on the private network can use the gateway's Internet address to connect to Internet hosts, you create a NAT table (**`-t nat`**) POSTROUTING rule with a MASQUERADE target.

```
iptables -t nat -A POSTROUTING -o eth0 -j MASQUERADE
```

Controlling ICMP Packets

In addition, to allow ping and destination-reachable ICMP packets, you enter INPUT rules with the firewall as the destination. To enable ping operations, you use both echo-reply and echo-request ICMP types, and for destination unreachable, you use the destination-unreachable type.

```
iptables -A INPUT -j ACCEPT  -p icmp -i eth0 --icmp-type \
   echo-reply -d 10.0.0.1
iptables -A INPUT -j ACCEPT  -p icmp -i eth0 --icmp-type \
   echo-request -d 10.0.0.1
iptables -A INPUT -j ACCEPT -p icmp -i eth0 --icmp-type \
   destination-unreachable -d 10.0.0.1
```

At the end, IP forwarding is turned on again.

```
echo 1 > /proc/sys/net/ipv4/ip_forward
```

Listing Rules

A listing of the **iptables** options shows the different rules for each option, as shown here:

```
# iptables -L
Chain INPUT (policy DROP)
target   prot opt source          destination
LOG      all  -- !192.168.0.0/24  anywhere        LOG level warning
DROP     all  -- !192.168.0.0/24  anywhere
DROP     all  --  192.168.0.0/24  anywhere
DROP     all  --  127.0.0.0/8     anywhere
ACCEPT   all  --  anywhere        anywhere
ACCEPT   tcp  --  10.0.0.2        anywhere        tcp dpt:http
ACCEPT   tcp  --  10.0.0.2        192.168.0.0/24  state RELATED,ESTABLISHED
                                                          tcp spt:http
ACCEPT  !icmp --  anywhere        anywhere        state RELATED,ESTABLISHED
DROP     all  --  anywhere        anywhere        state NEW
ACCEPT   all  --  192.168.0.0/24  anywhere
ACCEPT   icmp --  anywhere        10.0.0.1        icmp echo-reply
ACCEPT   icmp --  anywhere        10.0.0.1        icmp echo-request
ACCEPT   icmp --  anywhere        10.0.0.1        icmp destination-unreachable
Chain FORWARD (policy ACCEPT)
target   prot opt source          destination
DROP     all  -- !192.168.0.0/24  anywhere
DROP     all  --  192.168.0.0/24  anywhere
DROP     all  --  127.0.0.0/8     anywhere
DROP     all  --  anywhere        anywhere        state NEW

Chain OUTPUT (policy ACCEPT)
target   prot opt source       destination
DROP       tcp  --  anywhere     192.168.0.0/24  state NEW tcp spt:http

# iptables -t nat -L
Chain PREROUTING (policy ACCEPT)
target      prot opt source       destination
Chain POSTROUTING (policy ACCEPT)
target      prot opt source       destination
MASQUERADE  all  --  anywhere      anywhere
Chain OUTPUT (policy ACCEPT)
target     prot opt source       destination
```

User-Defined Rules

For more complex rules, you may want to create your own chain to reduce repetition. A common method is to define a user chain for both INPUT and FORWARD chains, so that you do not have to repeat DROP operations for each. Instead, you have only one user chain that both FORWARD and INPUT chains feed into for DROP operations. Keep in mind that both FORWARD and INPUT operations may have separate rules in addition to the ones they share.

In the next example, a user-defined chain called arriving is created. The chain is defined with the **-N** option at the top of the script:

```
iptables -N arriving
```

A user chain has to be defined before it can be used as a target in other rules, so you have to first define and add all the rules for that chain, and then use it as a target. The arriving chain is first defined and its rules added. Then, at the end of the file, it is used as a target for both the INPUT and FORWARD chains. The INPUT chain lists rules for accepting packets, whereas the FORWARD chain has an ACCEPT policy that will accept them by default.

```
iptables -N arriving
iptables -F arriving
# IP spoofing, deny any packets on the internal network
# that have an external source address.
iptables -A arriving -j LOG  -i eth1 \! -s 192.168.0.0/24
iptables -A arriving -j DROP  -i eth1 \! -s 192.168.0.0/24
iptables -A arriving -j DROP \! -i eth1 -s 192.168.0.0/24
.....
# entries at end of script
iptables -A INPUT -j arriving
iptables -A FORWARD -j arriving
```

A listing of the corresponding rules is shown here:

```
Chain INPUT (policy DROP)
target    prot opt source          destination
arriving  all  --  0.0.0.0/0         0.0.0.0/0
Chain FORWARD (policy ACCEPT)
target    prot opt source          destination
arriving  all  --  0.0.0.0/0         0.0.0.0/0
Chain arriving (2 references)
target    prot opt source          destination
LOG       all  -- !192.168.0.0/24   0.0.0.0/0       LOG flags 0 level 4
DROP      all  -- !192.168.0.0/24   0.0.0.0/0
DROP      all  --  192.168.0.0/24   0.0.0.0/0
```

For rules where chains may differ, you will still need to enter separate rules. In the **myfilter** script, the FORWARD chain has an ACCEPT policy, allowing all forwarded packets to the local network to pass through the firewall. If the FORWARD chain had a DROP policy, like the INPUT chain, then you may need to define separate rules under which the FORWARD chain could accept packets. In this example, the FORWARD and INPUT chains have different rules for accepting packets on the **eth1** device. The INPUT rule is more restrictive. To enable the local network to receive forwarded packets through the firewall, you can enable forwarding on its device using a separate FORWARD rule, as shown here:

```
iptables -A FORWARD -j ACCEPT -p all -i eth1
```

The INPUT chain will accept packets only from the local network.

```
iptables -A INPUT -j ACCEPT -p all -i eth1 -s 192.168.0.0/24
```

Simple LAN Configuration

To create a script to support a simple local area network (LAN) without any Internet services like web servers, you just don't include rules for supporting those services. You would still need FORWARD and POSTROUTING rules for connecting your localhosts to the Internet, as well as rules governing interaction between the hosts and the firewall. To modify the example script to support a simple LAN without the web server, just remove the three rules governing the web server. Leave everything else the same.

LAN Configuration with Internet Services on the Firewall System

Often, the same system that functions as a firewall is also used to run Internet servers such as web and FTP servers. In this case the firewall rules are applied to the ports used for those services. The example script dealt with a web server running on a separate host system. If the web server were instead running on the firewall system, you would apply the web server firewall rules to the port that the web server uses. Normally the port used for a web server is 80. In the following example, the IPtables rules for the web server have been applied to port www, port 80, on the firewall system. The modification simply requires removing the old web server host address references, 10.0.0.2.

```
# allow  communication to the web server, port www (port 80)
iptables -A INPUT  -j ACCEPT -p tcp -i eth0  --dport www
# Allow  established connections from web servers to internal network
iptables -A INPUT -m state --state ESTABLISHED,RELATED -i eth0 \
   -p tcp  --sport www -d 192.168.0.0/24  -j ACCEPT
# Prevent new  connections from web servers to internal network
iptables -A OUTPUT -m state --state  NEW -o eth0 -p tcp \
  --sport www -d 192.168.0.1.0/24 -j DROP
```

Similar entries can be set up for an FTP server. Should you run several Internet services, you could use a user-defined rule to run the same rules on each service, rather than repeating three separate rules per service. Working from the example script, you would use two defined rules, one for INPUT and one for OUTPUT, controlling incoming and outgoing packets for the services.

```
iptables -N inputservice
iptables -N outputservice
iptables -F inputservice
iptables -F outputservice
# allow  communication to the service
iptables -A inputservice  -j ACCEPT -p tcp -i eth0
# Allow  established connections from the service to internal network
iptables -A inputservice -m state --state ESTABLISHED,RELATED -i eth0 \
   -p tcp  -d 192.168.0.0/24  -j ACCEPT
# Prevent new  connections from service to internal network
iptables -A outputservice -m state --state  NEW -o eth0 -p tcp \
  -d 192.168.0.1.0/24 -j DROP
.....
# Run rules for the web server, port www (port 80)
iptables -A INPUT  --dport www -j inputservice
iptables -A INPUT  --dport www -j outputservice
# Run rules for the FTP server, port ftp (port 21)
iptables -A OUTPUT  --dport ftp -j inputservice
iptables -A OUTPUT  --dport ftp -j outputservice
```

IP Masquerading

On Linux systems, you can set up a network in which you can have one connection to the Internet, which several systems on your network can use. This way, using only one IP address, several different systems can connect to the Internet. This method is called *IP masquerading,* where a system masquerades as another system, using that system's IP address. In such a network, one system is connected to the Internet with its own IP address, while the other systems are connected on a LAN to this system. When a local system wants to access the network, it masquerades as the Internet-connected system, borrowing its IP address.

IP masquerading is implemented on Linux using the IPtables firewalling tool. In effect, you set up a firewall, which you then configure to do IP masquerading. Currently, IP masquerading supports all the common network services—as does IPtables firewalling—such as web browsing, Telnet, and ping. Other services, such as IRC, FTP, and RealAudio, require the use of certain modules. Any services you want local systems to access must also be on the firewall system because request and response are handled by services on that system.

You can find out more information on IP masquerading at the IP Masquerade Resource website at **ipmasq.webhop.net**. In particular, the Linux IP Masquerade mini-HOWTO provides a detailed, step-by-step guide to setting up IP masquerading on your system. IP masquerading must be supported by the kernel before you can use it. If your kernel does not support it, you may have to rebuild the kernel, including IP masquerade support, or use loadable modules to add it. See the IP Masquerade mini-HOWTO for more information.

With IP masquerading, as implemented on Linux systems, the machine with the Internet address is also the firewall and gateway for the LAN of machines that use the firewall's Internet address to connect to the Internet. Firewalls that also implement IP masquerading are sometimes referred to as *MASQ gates.* With IP masquerading, the Internet-connected system (the firewall) listens for Internet requests from hosts on its LAN. When it receives one, it replaces the requesting localhosts' IP address with the Internet IP address of the firewall and then passes the request out to the Internet, as if the request were its own. Replies from the Internet are then sent to the firewall system. The replies the firewall receives are addressed to the firewall using its Internet address. The firewall then determines the local system to whose request the reply is responding. It then strips off its IP address and sends the response on to the localhost across the LAN. The connection is transparent from the perspective of the local machines. They appear to be connected directly to the Internet.

Masquerading Local Networks

IP masquerading is often used to allow machines on a private network to access the Internet. These could be machines in a home network or a small LAN, such as for a small business. Such a network might have only one machine with Internet access, and as such, only the one Internet address. The local private network would have IP addresses chosen from the private network allocations (10., 172.16., or 192.168.). Ideally, the firewall has two Ethernet cards: one for an interface to the LAN (for example, **eth1**) and one for an interface to the Internet, such as **eth0** (for dial-up ISPs, this would be **ppp0** for the modem). The card for the Internet connection (**eth0**) would be assigned the Internet IP address. The Ethernet interface for the local network (**eth1**, in this example) is the firewall Ethernet interface. Your private LAN would have a network address like 192.168.0. Its Ethernet firewall interface (**eth1**) would be assigned the IP address 192.168.0.1. In effect, the firewall interface lets the firewall operate as

the local network's gateway. The firewall is then configured to masquerade any packets coming from the private network. Your LAN needs to have its own domain name server, identifying the machines on your network, including your firewall. Each local machine needs to have the firewall specified as its gateway. Try not to use IP aliasing to assign both the firewall and Internet IP addresses to the same physical interface. Use separate interfaces for them, such as two Ethernet cards, or an Ethernet card and a modem (**ppp0**).

Masquerading NAT Rules

In Netfilter, IP masquerading is a NAT operation and is not integrated with packet filtering as in IP Chains. IP masquerading commands are placed on the NAT table and treated separately from the packet-filtering commands. Use IPtables to place a masquerade rule on the NAT table. First reference the NAT table with the **-t nat** option. Then add a rule to the POSTROUTING chain with the **-o** option specifying the output device and the **-j** option with the MASQUERADE command.

```
iptables -t nat -A POSTROUTING -o eth0 -j MASQUERADE
```

IP Forwarding

The next step is to turn on IP forwarding, either manually or by setting the **net.ipv4.ip_forward** variable in the **/etc/sysctl.conf** file and running **sysctl** with the **-p** option. IP forwarding will be turned off by default. For IPv6, use **net.ipv6.conf.all.forwarding**. The **/etc/sysctl.conf** entries are shown here:

```
net.ipv4.ip_forward = 1
net.ipv6.conf.all.forwarding = 1
```

You then run **sysctl** with the **-p** option.

```
sysctl -p
```

You can directly change the respective forwarding files with an **echo** command as shown here:

```
echo 1 > /proc/sys/net/ipv4/ip_forward
```

For IPv6, you use the forwarding file in the corresponding **/proc/sys/net/ipv6** directory, **conf/all/forwarding**.

```
echo 1 > /proc/sys/net/ipv6/conf/all/forwarding
```

Masquerading Selected Hosts

Instead of masquerading all localhosts as the single IP address of the firewall/gateway host, you can use the NAT table to rewrite addresses for a few selected hosts. Such an approach is often applied to setups where you want several localhosts to appear as Internet servers. Using the DNAT and SNAT targets, you can direct packets to specific localhosts. You use rules on the PREROUTING and POSTROUTING chains to direct input and output packets.

For example, the web server described in the previous example could have been configured as a localhost to which a DNAT target could redirect any packets originally

received for 10.0.0.2. Say the web server was set up on 192.168.0.5. It could appear as having the address 10.0.0.2 on the Internet. Packets sent to 10.0.0.2 would be rewritten and directed to 192.168.0.5 by the NAT table. You would use the PREROUTING chain with the **-d** option to handle incoming packets and POSTROUTING with the **-s** option for outgoing packets.

```
iptables -t nat -A PREROUTING -d 10.0.0.2  \
           --to-destination 192.168.0.5 -j DNAT
iptables -t nat -A POSTROUTING -s 192.168.0.5 \
           --to-source 10.0.0.2 -j SNAT
```

TIP *Bear in mind that with IPtables, masquerading is not combined with the FORWARD chain, as it is with IP Chains. So, if you specify a DROP policy for the FORWARD chain, you will also have to specifically enable FORWARD operation for the network that is being masqueraded. You will need both a POSTROUTING rule and a FORWARD rule.*

This page intentionally left blank

PART VI

Internet and Network Services

CHAPTER 21
Managing Services

CHAPTER 22
FTP Server

CHAPTER 23
Web Servers

CHAPTER 24
Proxy Servers

CHAPTER 25
Mail Servers

CHAPTER 26
Print, News, Search, and Database Servers

This page intentionally left blank

21 CHAPTER

Managing Services

A single Linux system can provide several different kinds of services, ranging from security to administration and including more obvious Internet services like websites and FTP sites, email, and printing. Security tools such as SSH and Kerberos run as services, along with administrative network tools such as DHCP and LDAP. The network connection interface is itself a service that you can restart at will. Each service operates as a continually running daemon looking for requests for its particular services. In the case of a web service, the requests will come from remote users. You can turn services on or off by starting or shutting down their daemons.

The process of starting up or shutting down a service is handled by service scripts, described in detail in this chapter. It applies to all services, including those discussed in the "Security', "System Administration", and "Network Administration" parts of this book. It is covered at this point since you will most likely use them to start and stop Internet services like web and mail servers.

System Files: /etc/rc.d

Each time you start your system, it reads a series of startup commands from system initialization files located in your **/etc/rc.d** directory. These initialization files are organized according to different tasks. Some are located in the **/etc/rc.d** directory itself, while others are located in a subdirectory called **init.d**. You should not have to change any of these files. The organization of system initialization files varies among Linux distributions. Some of the files you find in **/etc/rc.d** are listed in Table 21-1.

rc.sysinit and rc.local

The **/etc/rc.d/rc.sysinit** file holds the commands for initializing your system, including the mounting and unmounting of your file systems. The **/etc/rc.d/rc.local** file is the last initialization file executed. You can place commands of your own here. When you shut down your system, the system calls the **halt** file, which contains shutdown commands. The files in **init.d** are then called to shut down daemons, and the file systems are unmounted. **halt** is located in the **init.d** directory.

Files and Directories	Description
/etc/sysconfig	Directory that holds system configuration files and directories.
/etc/rc.d	Directory that holds system startup and shutdown files.
/etc/rc.d/rc.sysinit	Initialization file for your system.
/etc/init.d/rc.local	Initialization file for your own commands; you can freely edit this file to add your own startup commands; this is the last startup file executed.
/etc/init.d	Directory that holds network scripts to start up network connections.
/etc/rc.d/rc*num***.d**	Directories for different runlevels, where *num* is the runlevel. The directories hold links to scripts in the **/etc/init.d** directory.
/etc/init.d	Directory that holds system service scripts (see Table 21-2).
/etc/init.d/halt	Operations performed each time you shut down the system, such as unmounting file systems; called rc.halt in other distributions.

TABLE 21-1 System Files and Directories

/etc//init.d

The **/etc/init.d** directory is designed primarily to hold scripts that start up and shut down different specialized daemons, such as network and printer daemons and those for font and web servers. These files perform double duty, starting a daemon when the system starts up and shutting down the daemon when the system shuts down. The files in **init.d** are designed in a way to make it easy to write scripts for starting up and shutting down specialized applications. They use functions defined in the **functions** file. Many of these files are set up for you automatically. You shouldn't need to change them. If you do change them, first be sure you know how these files work.

When your system starts up, several programs are automatically started and run continuously to provide services, such as a website or print servers. Depending on what kind of services you want your system to provide, you can add or remove items in a list of services to be started automatically. For example, the web server is run automatically when your system starts up. If you are not hosting a website, you have no need for the web server. You can prevent the service from starting, removing an extra task the system does not need to perform, freeing up resources, and possibly reducing potential security holes. Several of the servers and daemons perform necessary tasks. The **sendmail** server enables you to send messages across networks, and the **cupsd** server performs printing operations.

To configure a service to start up automatically at boot, you can use a distribution startup configuration tools. Red Hat and similar distributions like SUSE and Fedora use **chkconfig**, whereas Debian and its similar distributions like Ubuntu use **rrconf** or **sysv-rc-conf** tools. The **chkconfig** command uses the **on** and **off** options to select and deselect services for startup at boot (see later in this chapter).

```
chkconfig httpd on
```

For Linux distributions that support SysV Init scripts, you can use the **service** command to start and stop services manually at any time. With the **service** command, you list the service with the **stop** argument to stop it, the **start** argument to start it, and the **restart** argument to restart it.

```
service httpd start
```

NOTE *On Debian and similar distributions, you may have to manually install the* **service** *tool. The names for service scripts may differ, like* ***apache2*** *instead of* ***httpd*** *for the web server.*

SysV Init: init.d Scripts

You can manage the startup and shutdown of server daemons with special service scripts located in the **/etc/init.d** directory. These scripts often have the same name as the service's program. For example, for the **/usr/sbin/httpd** web server program, the corresponding script is called **/etc/init.d/httpd**. This script starts and stops the web server. This method of using **init.d** service scripts to start servers is called *SysV Init*, after the method used in Unix System V. Some of the more commonly used service scripts are listed in Table 21-2.

The service scripts in the **/etc/init.d** directory can be executed automatically whenever you boot your system. Be careful when accessing these scripts, however. These start essential programs, such as your network interface and your printer daemon. These init scripts are accessed from links in subdirectories set up for each possible runlevel. The **/etc/rc.d** directory holds a set of subdirectories whose names have the format **rc***n***.d**, where *n* is a number referring to a runlevel (there are also links in the **/etc** directory directly to the **/etc/rc.d** runlevel subdirectories). Runlevel tasks will differ according to distribution, though most distributions use either the Red Hat or Debian designations (see Table 21-3).

The **rc** script detects the runlevel in which the system was started and then executes only the service scripts specified in the subdirectory for that runlevel. When you start your system, the **rc** script executes the service scripts designated default startup directory like in **rc5.d** (graphical login for Fedora and SUSE) or **rc1.d** (graphical login for Debian and Ubuntu). Sone distributions will have a command line login like rc3.d on Fedora, Red Hat, and SUSE. The **rc***n***.d** directories hold symbolic links to certain service scripts in the **/etc/init.d** directory. Thus, the **httpd** script in the **/etc/init.d** directory is actually called through a symbolic link in an **rcn.d** directory. The symbolic link for the **/etc/init.d/httpd** script in the **rc3.d** directory on Fedora is **S85httpd**. The *S* prefixing the link stands for "startup"; thus, the link calls the corresponding **init.d** script with the **start** option. The number indicates the order in which service scripts are run; lower numbers run first. **S85httpd** invokes **/etc/init.d/httpd** with the option **start**. If you change the name of the link to start with a *K*, the script is invoked with the **stop** option, stopping it. Such links are used in the runlevels 0 and 6 directories, **rc6.d** and **rc0.d**. Runlevel 0 halts the system, and runlevel 6 reboots it. You can use the **runlevel** command to find out what runlevel you are currently operating at (see Chapter 27 for more details on runlevels). A listing of runlevels is shown in Table 21-3.

Service Script	Description
`network`	Operations to start up or shut down your network connections
`xinetd`	Operations to start up or shut down the **xinetd** daemon
`autofs`	Automatic file system mounting (see Chapter 29)
`cups`	The CUPS printer daemon (see Chapter 25)
`cpuspeed`	Service to manage CPU speed (Athlon Cool and Quiet)
`dhcpd`	Dynamic Host Configuration Protocol daemon (see Chapter 36)
`httpd`	Apache web server (**apache2** on Debian, Chapter 23)
`innd`	Internet News service (see Chapter 26)
`ipsec`	IPsec secure VPN service (see Chapter 18)
`iptables`	Controls the IPtables daemon
`ip6tables`	IPtables for IP protocol version 6 (see Chapter 20)
`krb5kdc`	Kerberos kdc server (see Chapter 19)
`ldap`	LDAP service (see Chapter 28)
`nfs`	Network file system (see Chapter 37)
`postfix`	Postfix mail server (see Chapter 25)
`sendmail`	The Sendmail MTA daemon (see Chapter 25)
`smb`	Samba for Windows hosts (**samba** on Debian)
`squid`	Squid proxy-cache server (see Chapter 24)
`sshd`	Secure Shell daemon (see Chapter 19)
`syslog`	System logging daemon (see Chapter 27)
`vsftpd`	Very secure FTP server (see Chapter 22)
`ypbind`	Network Information Service (NIS) (see Chapter 37)

TABLE 21-2 Selection of Service Scripts in /etc/init.d

Starting Services: Standalone and xinetd

A *service* is a daemon that runs concurrently with your other programs, continually looking for a request for its services, either from other users on your system or from remote users connecting to your system through a network. When a server receives a request from a user, it starts up a *session* to provide its services. For example, if users want to download a file from your system, they can use their own FTP client to connect to your FTP server and start up a session. In the session, they can access and download files from your system. Your server needs to be running for a user to access its services. For example, if you set up a website on your system with HTML files, you must have the **httpd** web server program running before users can access your website and display those files.

Runlevel	rc.d Directory	Description
Red Hat		**Red Hat, Fedora, SUSE, and associated distributions**
0	**rc0.d**	Halts (shuts down) the system
1	**rc1.d**	Single-user mode (no networking, limited capabilities)
2	**rc2.d**	Multiuser mode with no NFS support (limited capabilities)
3	**rc3.d**	Multiuser mode (full operational mode, no graphical login, command line interface is default)
4	**rc4.d**	User-defined
5	**rc5.d**	Multiuser mode with graphical login (full operation mode with graphical login and graphical interface as default)
6	**rc6.d**	Reboots system
S	**rcS.d**	Single-user mode
Debian		**Debian, Ubuntu, and associated distributions**
0	**rc0.d**	Halts (shuts down) the system
1	**rc1.d**	Single-user mode (limited capabilities)
2	**rc2.d**	Multiuser mode with graphical login (full operation mode, X server started automatically)
3	**rc3.d**	User-defined
4	**rc4.d**	User-defined
5	**rc5.d**	User-defined
6	**rc6.d**	Reboots system
S	**rcS.d**	Single-user mode

TABLE 21-3 System Runlevels for Red Hat and Debian distributions

Starting Services Directly

You can start a server in several ways. One way is to do it manually from the command line by entering the name of the server program and its arguments. When you press ENTER, the server starts, although your command line prompt reappears. The server runs concurrently as you perform other tasks. To see if your server is running, you can use the **service** command with the **status** option.

```
# service httpd status
```

Alternatively, you can use the **ps** command with the **-aux** option to list all currently running processes. You should see a process for the server program you started. To refine the list, you can add a **grep** operation with a pattern for the server name you want. The second command lists the process for the web server.

```
# ps -aux
# ps -aux | grep 'httpd'
```

You can just as easily check for the **httpd** process on the GNOME System Monitor.

Starting and Stopping Services with Service Scripts

On distributions that support SysV Init scripts, you can use service scripts to start and stop your server manually. These scripts are located in the **/etc/init.d** directory and have the same names as the server programs. For example, the **/etc/init.d/httpd** script with the **start** option starts the web server. Using this script with the **stop** option stops it. Instead of using the complete pathname for the script, you can use the **service** command and the script name. The following commands are equivalent:

```
/etc/init.d/httpd stop
service httpd stop
```

Starting Services Automatically

Instead of manually executing all the server programs each time you boot your system, you can have your system automatically start the servers for you. You can do this in two ways, depending on how you want to use a server. You can have a server running continuously from the time you start your system until you shut it down, or you can have the server start only when it receives a request from a user for its services. If a server is being used frequently, you may want to have it running all the time. If it is used rarely, you may want the server to start only when it receives a request. For example, if you are hosting a website, your web server is receiving requests all the time from remote users on the Internet. For an FTP site, however, you may receive requests infrequently, in which case you may want to have the FTP server start only when it receives a request. Of course, certain FTP sites receive frequent requests, which would warrant a continuously running FTP server.

Standalone Servers

A server that starts automatically and runs continuously is referred to as a *standalone* server. The SysV Init procedure can be used to start servers automatically whenever your system boots. This procedure uses service scripts for the servers located in the **/etc/init.d** directory. Most Linux systems configure the web server to start automatically and to run continuously by default. A script for it called **httpd** is in the **/etc/init.d** directory.

xinetd Servers

To start the server only when a request for its services is received, you configure it using the **xinetd** daemon. If you add, change, or delete server entries in the **/etc/xinetd** files, you will have to restart the **xinetd** daemon for these changes to take effect. On distributions that support SysV Init scripts, you can restart the **xinetd** daemon using the **/etc/init.d/xinetd** script with the **restart** argument, as shown here:

```
# service xinetd restart
```

You can also use the **xinetd** script to start and stop the **xinetd** daemon. Stopping effectively shuts down all the servers that the **xinetd** daemon manages (those listed in the **/etc/xinetd.conf** file or the **xinetd.d** directory).

```
# service xinetd stop
# service xinetd start
```

You can also directly restart **xinetd** by stopping its process directly. To do this, you use the **killall** command with the **-HUP** signal and the name **xinetd**.

```
# killall -HUP xinetd
```

Service Management: chkconfig, services-admin, rrconf, sysv-rc-conf, and update-rc.d

Though there is no distribution-independent tool for managing servers, most distributions use the **chkconfig** ,**services-admin** (GNOME), **rrconf** (Debian), **sysv-rc-conf**, or **update-rc.d** tools. The **chkconfig** tool was developed by Red Hat and is used on Red Hat, Fedora, and SUSE, and similar distributions, whereas **rrconf** and **update-rc.d** were developed by Debian and are used on Debian, Ubuntu, and similar distributions. The **sysv-rc-conf** tool is a generic tool that can be used on all distributions. The **services-admin** tool is part of GNOME system tools and is available for all distributions.

The tools provide simple interfaces you can use to choose what servers you want started up and how you want them to run. You use these tools to control any daemon you want started up, including system services such as **cron**, the print server, remote file servers for Samba and NFS, authentication servers for Kerberos, and, of course, Internet servers for FTP or HTTP. Such daemons are referred to as *services*, and you should think of these tools as managing these services. Any of these services can be set up to start or stop at different runlevels.

If you add a new service, **chkconfig**, **services-admin**, **rcconf**, or **sysv-rc-conf** can manage it. As described in the following section, services are started up at specific runlevels using service links in various runlevel directories. These links are connected to the service scripts in the **init.d** directory. Runlevel directories are numbered from 0 to 6 in the **/etc/rc.d** directory, such as **/etc/rc.d/rc3.d** for runlevel 3 and **/etc/rc.d/rc5.d** for runlevel 5. Removing a service from a runlevel only changes its link in the corresponding runlevel **rc.d** directory. It does not touch the service script in the **init.d** directory.

chkconfig

You can specify the service you want to start and the level you want to start it at with the **chkconfig** command. Unlike other service management tools, **chkconfig** works equally well on standalone and **xinetd** services. Though standalone services can be run at any runlevel, you can also turn **xinetd** services on or off for the runlevels that **xinetd** runs in. Table 21-4 lists the different **chkconfig** options.

Listing Services with chkconfig

To see a list of services, use the **--list** option. A sampling of services managed by **chkconfig** is shown here. The on or off status of the service is shown at each runlevel, as are **xinetd** services and their statuses:

```
chkconfig -list
dhcpd   0:off 1:off 2:off 3:off 4:off 5:off 6:off
httpd   0:off 1:off 2:off 3:off 4:off 5:off 6:off
named   0:off 1:off 2:off 3:off 4:off 5:off 6:off
lpd     0:off 1:off 2:on  3:on  4:on  5:on  6:off
```

Option	Description
`--level` *runlevel*	Specifies a runlevel to turn on, turn off, or reset a service.
`--list` *service*	Lists startup information for services at different runlevels. **xinetd** services are just `on` or `off`. With no argument, all services are listed, including **xinetd** services.
`--add` *service*	Adds a service, creating links in the default specified runlevels (or all runlevels, if none are specified).
`--del` *service*	Deletes all links for the service (startup and shutdown) in all runlevel directories.
service `on`	Turns a service on, creating a service link in the specified or default runlevel directories.
service `off`	Turns a service off, creating shutdown links in specified or default directories.
service `reset`	Resets service startup information, creating default links as specified in the `chkconfig` entry in the service's `init.d` service script.

TABLE 21-4 Options for chkconfig

```
nfs     0:off 1:off 2:off 3:off 4:off 5:off 6:off
crond   0:off 1:off 2:on  3:on  4:on  5:on  6:off
xinetd 0:off 1:off 2:off 3:on  4:on  5:on  6:off
xinetd based services:
     time:       off
     finger:     off
     pop3s:      off
     swat:       on
```

Starting and Stopping Services with chkconfig

You use the `on` option to have a service started at specified runlevels and the `off` option to disable it. You can specify the runlevel to affect with the `--level` option. If no level is specified, `chkconfig` will use any `chkconfig` default information in a service's `init.d` service script. Distributions usually install their services with `chkconfig` default information already entered (if this is missing, `chkconfig` will use runlevels 3, 4, and 5). The following example has the web server (**httpd**) started at runlevel 5:

```
chkconfig --level 5 httpd on
```

The `off` option configures a service to shut down if the system enters a specified runlevel. The next example shuts down the web server if runlevel 3 is entered. If the service is not running, it remains shut down:

```
chkconfig --level 3 httpd off
```

The `reset` option restores a service to its `chkconfig` default options as specified in the service's `init.d` service script:

```
chkconfig httpd reset
```

To see just the startup information for a service, you use just the service name with the **--list** option:

```
chkconfig --list httpd
httpd   0:off  1:off 2:off 3:on 4:off 5:on 6:off
```

Enabling and Disabling xinetd Services with chkconfig

The **chkconfig** command can also enable or disable **xinetd** services. Simply enter the **xinetd** service with either an **on** or **off** option. The service will be started up or shut down, and the disable line in its **xinetd** configuration script in the **/etc/xinetd.d** directory will be edited accordingly. For example, to start SWAT, the Samba configuration server, which runs on **xinetd**, you simply enter

```
chkconfig swat on
chkconfig --list swat
     swat           on
```

The **swat** configuration file for **xinetd**, **/etc/xinetd.d/swat**, will have its disable line edited to no, as shown here:

```
disable=no
```

If you want to shut down the SWAT server, you can use the **off** option. This will change the disable line in **/etc/xinetd.d/swat** to read "disable=yes".

```
chkconfig swat off
```

The same procedure works for other **xinetd** services such as the POP3 server and **finger**.

Removing and Adding Services with chkconfig

If you want a service removed entirely from the entire startup and shutdown process in all runlevels, you can use the **--del** option. This removes all startup and shutdown links in all the runlevel directories.

```
chkconfig --del httpd
```

You can also add services to **chkconfig** management with the **--add** option; **chkconfig** will create startup links for the new service in the appropriate startup directories, **/etc/rc.d/rc***n***.d**. If you have previously removed all links for a service, you can restore them with the **add** option.

```
chkconfig --add httpd
```

Configuring xinetd Services for Use by chkconfig

Default runlevel information should be placed in the service scripts that are to be managed by **chkconfig**. You can edit these scripts to change the default information if you wish. This information is entered as a line beginning with a **#** sign and followed by the **chkconfig** keyword and a colon. Then you list the default runlevels that the service should start up on, along with the start and stop priorities. The following entry lists runlevels 3 and 5 with

a start priority of 85 and a stop of 15. See the section "Service Script Tags" for more information:

```
# chkconfig: 35 85 15
```

Thus, when a user turns on the **httpd** service with no level option specified, **chkconfig** will start up **httpd** at runlevels 3 and 5.

```
chkconfig httpd on
```

How chkconfig Works

The **chkconfig** tool works by creating startup and shutdown links in the appropriate runlevel directories in the **/etc/rc.d** directory. For example, when **chkconfig** adds the **httpd** service at runlevel 5, it creates a link in the **/etc/rc.d/rc5.d** directory to the service script **httpd** in the **/etc/init.d** directory. When it turns off the web service from runlevel 3, it creates a shutdown link in the **/etc/rc.d/rc3.d** directory to use the script **httpd** in the **/etc/init.d** directory to make sure the web service is not started. In the following example, the user turns on the web service (**httpd**) on runlevel 3, creating the startup link in **rc5.d**, **S85httpd**, and then turns off the web service on runlevel 3, creating a shutdown link in **rc3.d**, **K15httpd**.

```
chkconfig --level 5 httpd on
ls /etc/rc.d/rc5.d/*httpd
   /etc/rc.d/rc5.d/S85httpd
chkconfig -level 3 httpd off
ls /etc/rc.d/rc3.d/*httpd
  /etc/rc.d/rc3.d/K15httpd
```

rcconf, services-admin, sysv-rc-conf, and update-rc.d

On Debian and similar systems, to turn services on or off for different runlevels, like **chkconfig** can, you should use **rcconf** or **sysv-rc-conf**. Both tools are run from a terminal window on the command line. Both provide an easy cursor-based interface for using arrow keys and the spacebar to turn services on or off. The rcconf tool is a more limited tool that turns services on or off for all the default runlevels, whereas sysv-rc-conf is more refined, allowing you to select specific runlevels.

GNOME's services-admin tool is a very limited tool like rcconf, which simply turns services on or off, withouth specifying any runlevels. It provides a GUI on GNOME, usually accessible from the System | Administration menu as the Services entry. Every service has a checkbox. Those checked will be started at boot time, those unchecked will not. To turn on a service, scroll to its entry and click the checkbox next to it, if empty. To turn off a service, click its checkbox again.

NOTE *The Boot Up Manager* ***(bum)*** *provides a simple desktop interface for turning services on and off. Its features are similar to* ***rrconf.***

The **sysv-rc-conf** tool displays a cursor-based screen where you can check which services to run or stop, and at which runlevel. The runlevels will be listed from 0 to 6 and S.

Use the arrow keys to position to the cell for your service and runlevel. Then use the spacebar to turn a service on or off.

The **update-rc.d** tool is a lower-level tool that can install or remove runlevel links. It is usually used when installaing service packages to create default runlevel links. You can use it to configure your own runlevels for a service, but requires detailed understanding of how runlevel links for services are configured.

The **update-rc.d** tool does not affect links that are already installed. It only works on links that are not already present in the runlevel directories. In this respect, it cannot turn a service on or off directly like **sysv-rc-conf** and **chkconfig** can. To turn off a service you would first have to remove all runlevel links in all the rc*n*.d directories using the **remove** option, and then add in the ones you want with the **start** or **stop** options. This makes turning services on an off using the **update-rc.d** tool much more complicated.

You use **start** and **stop** options along with the runlevel to set the runlevels at which to start or stop a service. You will need to provide a link number for ordering the sequence in which it will be run. You then enter the runlevel followed by a period. You can specify more than one runlevel. The following will start the web server on runlevel 5. The order number used for the link name is 91. The link name will be **S91apache**.

```
update-rc.d apache start 91 5 .
```

The stop number is always 100 minus the start number. So the stop number for service with a start number of 91 would be 09.

```
update-rc.d apache stop 09 6 .
```

The **start** and **stop** options can be combined.

```
update-rc.d apache 99 start 5 . stop 09 6 .
```

A **defaults** option will start and stop the service at a predetermined runlevel. This option can be used to quickly set standard start and stop links for all runlevels. Startup links will be set in runlevels 2, 3, 4, and 5. Stop entries are set in runlevels 0, 1, and 6.

```
update-rc.d apache defaults
```

The following command performs the same operation with the stop and start options.

```
update-rc.d apache 99 start 2 3 4 5 . stop 09 0 1 6 .
```

The **multiuser** options will start entries at 2, 3, 4 ,5 and stop them at 1.

```
update-rc.d apache multiuser
```

To remove a service you use the remove option. The links will not be removed if the service script is still present in the **init.d** directory. Use the **-f** option to force removal of the links without having to remove the service script. The following removes all web service startup and shutdown entries from all runlevels.

```
update-rc.d –f apache  remove
```

To turn off a service at a given runlevel that is already turned on, you would have to first remove all its runlevel links and then add in the ones you want. So to turn off the Apache server at runlevel 3, but still have it turned on at runlevels 2, 4, and 5 you would use the following commands.

```
update-rc.d -f apache remove
update-rc.d apache 99 start 2 4 5 . stop 09 0 1 3 6 .
```

Keep in mind that the **remove** option removes all stop links as well as start ones. So you have to restore the stop links for 0, 1, and 6.

TIP *On Debian and Ubuntu you can use file-rc instead of sysv-rc. The file-rc tool uses a single configuration file instead of links in separate runlevel directories.*

Service Scripts: /etc/init.d

Most software using RPM packages will automatically install any appropriate service scripts and create the needed links in the appropriate **rc*n*.d** directories, where *n* is the runlevel number. Service scripts, though, can be used for any program you may want run when your system starts up. To have such a program start automatically, you first create a service script for it in the **/etc/init.d** directory and then create symbolic links to that script in the **/etc/rc.d/rc3.d** and **/etc/rc.d/rc5.d** directories. A shutdown link (*K*) should also be placed in the **rc6.d** directory used for runlevel 6 (reboot).

Service Script Functions

A simplified version of the service script **httpd** is shown in a later section. You can see the different options, listed in the **`/etc/init.d/httpd`** example, under the **`case`** statement: **`start`**, **`stop`**, **`status`**, **`restart`**, and **`reload`**. If no option is provided (**`*`**), the script-use syntax is displayed. The **`httpd`** script first executes a script to define functions used in these service scripts. The **`daemon`** function with **`httpd`** actually executes the **/usr/sbin/httpd** server program.

```
echo -n "Starting httpd: "
 daemon httpd
 echo
 touch /var/lock/subsys/httpd
```

The **`killproc`** function shuts down the daemon. The lock file and the process ID file (**httpd.pid**) are then deleted:

```
 killproc httpd
 echo
 rm -f /var/lock/subsys/httpd
 rm -f /var/run/httpd.pid
```

The **`daemon`**, **`killproc`**, and **`status`** functions are shell scripts defined in the **`functions`** script, also located in the **inet.d** directory. The **`functions`** script is executed at

Init Script Function	Description
daemon [+/-*nicelevel*] *program* [*arguments*] [**&**]	Starts a daemon, if it is not already running.
killproc *program* [*signal*]	Sends a signal to the program; by default it sends a **SIGTERM**, and if the process doesn't stop, it sends a **SIGKILL**. It will also remove any PID files, if it can.
pidofproc *program*	Used by another function, it determines the PID of a program.
status *program*	Displays status information.

TABLE 21-5 Init Script Functions

the beginning of each service script to activate these functions. A list of these functions is provided in Table 21-5.

```
. /etc/init.d/functions
```

Service Script Tags

The beginning of the service script holds tags used to configure the server. These tags, which begin with an initial **#** symbol, are used to provide runtime information about the service to your system. The tags are listed in Table 21-6, along with the service functions.

Init Script Tag	Description
# chkconfig: *startlevellist startpriority endpriority*	Required. Specifies the default start levels for this service as well as start and end priorities.
# description [*ln*]**:** *description of service*	Required. The description of the service, continued with \ characters. Use an initial **#** for any added lines. With the *ln* option, you can specify the language the description is written in.
# autoreload: *true*	Optional. If this line exists, the daemon checks its configuration files and reloads them automatically when they change.
# processname: *program*	Optional, multiple entries allowed. The name of the program or daemon started in the script.
# config: *configuration-file*	Optional, multiple entries allowed. Specifies a configuration file used by the server.
# pidfile: *pid-file*	Optional, multiple entries allowed. Specifies the PID file.
# probe: *true*	Optional, used *in place* of **autoreload**, **processname**, **config**, and **pidfile** entries to automatically probe and start the service.

TABLE 21-6 System V Init Script Tags

You enter a tag with a preceding **#** symbol, the tag name with a colon, and then the tag arguments. For example, the **processname** tag specifies the name of the program being executed, in this example **httpd**:

```
# processname: httpd
```

If your script starts more than one daemon, you should have a **processname** entry for each. For example, the Samba service starts up both the **smdb** and **nmdb** daemons.

```
# processname: smdb
# processname: nmdb
```

The end of the tag section is indicated by an empty line. After this line, any lines beginning with a **#** are treated as comments. The **chkconfig** line lists the default runlevels that the service should start up on, along with the start and stop priorities. The following entry lists runlevels 3, 4, and 5 with a start priority of 85 and a stop of 15:

```
# chkconfig: 345 85 15
```

For the description, you enter a short explanation of the service, using the \ symbol before a newline to use more than one line.

```
# description: Apache web server
```

With **config** tags, you specify the configuration files the server may use. In the case of the Apache web server, there may be three configuration files:

```
# config: /etc/httpd/conf/access.conf
# config: /etc/httpd/conf/httpd.conf
# config: /etc/httpd/conf/srm.conf
```

The **pidfile** entry indicates the file where the server's process ID is held.

Service Script Example

As an example, a simplified version of the web server service script, **/etc.init.d/httpd**, is shown here. Most scripts are much more complicated, particularly when determining any arguments or variables a server may need to specify when it starts up. This script has the same name as the web server daemon, **httpd**:

```
#!/bin/sh
#
# Service script for the Apache web Server
#
# chkconfig: 35 85 15
# description: Apache is a World Wide Web server. \
# It is used to serve HTML files and CGI.
# processname: httpd
# pidfile: /var/run/httpd.pid
# config: /etc/httpd/conf/access.conf
# config: /etc/httpd/conf/httpd.conf
# config: /etc/httpd/conf/srm.conf
```

```
# Source function library.
. /etc/init.d/functions

# See how we were called.
case "$1" in
 start)
  echo -n "Starting httpd: "
   daemon httpd
   echo
   touch /var/lock/subsys/httpd
   ;;
  stop)
   killproc httpd
   echo
   rm -f /var/lock/subsys/httpd
   rm -f /var/run/httpd.pid
   ;;
 status)
   status httpd
   ;;
 restart)
   $0 stop
   $0 start
   ;;
 reload)
   echo -n "Reloading httpd: "
   killproc httpd -HUP
   echo
   ;;
 *)
   echo "Usage: $0 {start|stop|restart}"
   exit 1
 esac
exit 0
```

Installing Service Scripts

The distributions that support SysV Init scripts will include a service script in their service package. For example, an Internet server package includes the service script for that server. Installing the RPM package installs the script in the **/etc/init.d** directory and creates its appropriate links in the runlevel directories, such as **/etc/rc.h/rc3.d**. If you decide, instead, to create the server using its source code files, you can then manually install the service script. If no service script exists, you first make a copy of the **`httpd`** script—renaming it—and then edit the copy to replace all references to **`httpd`** with the name of the server daemon program. Then place the copy of the script in the **/etc/init.d** directory and make a symbolic link to it in the **/etc/rc.d/rc3.d** directory.

Extended Internet Services Daemon (xinetd)

If your system averages only a few requests for a specific service, you don't need the server for that service running all the time. You need it only when a remote user is accessing its service. The Extended Internet Services Daemon (**xinetd**) manages Internet servers,

invoking them only when your system receives a request for their services. **xinetd** checks continuously for any requests by remote users for a particular Internet service; when it receives a request, it then starts the appropriate server daemon.

The **xinetd** program is designed to be a replacement for **inetd**, providing security enhancements, logging support, and even user notifications. For example, with **xinetd** you can send banner notices to users when they are not able to access a service, telling them why. **xinetd** security capabilities can be used to prevent denial-of-service attacks, limiting remote hosts' simultaneous connections or restricting the rate of incoming connections. **xinetd** also incorporates TCP, providing TCP security without the need to invoke the **tcpd** daemon. Furthermore, you do not have to have a service listed in the **/etc/services** file. **xinetd** can be set up to start any kind of special-purpose server.

Starting and Stopping xinetd Services

On Red Hat and associated distributions, you can turn on and off particular **xinetd** services with **chkconfig**, as described earlier. Use the **on** and **off** options to enable or disable a service; **chkconfig** will edit the disable option for the service, changing its value to "yes" for off and "no" for on. For example, to enable the SWAT server, you could enter

```
chkconfig swat on
```

For distributions that support SysV Init scripts, you can start, stop, and restart **xinetd** using its service script in the **/etc/init.d** directory, as shown here:

```
# service xinetd stop
# service xinetd start
# service xinetd restart
```

xinetd Configuration: xinetd.conf

The xinetd.conf file contains settings for your xinetd server, such as logging and security attributes (see Table 21-7 later in this chapter). This file can also contain server configuration entries, or they may be placed into separate configuration files located in the **/etc/xinetd.d** directory. The **includedir** attribute specifies this directory:

```
includedir /etc/xinetd.d
```

Logging xinetd Services

You can further add a variety of other attributes such as logging information about connections and server priority (**nice**). In the following example, the **log_on_success** attribute logs the duration (**DURATION**) and the user ID (**USERID**) for connections to a service, **log_on_failure** logs the users that failed to connect, and **nice** sets the priority of the service to 10.

```
log_on_success += DURATION USERID
log_on_failure += USERID
nice = 10
```

The default attributes defined in the defaults block often set global attributes such as default logging activity and security restrictions: **log_type** specifies where logging information is to be sent, such as to a specific file (**FILE**) or to the system logger (**SYSLOG**),

log_on_success specifies information to be logged when connections are made, and **log_on_failure** specifies information to be logged when they fail.

```
log_type = SYSLOG daemon info
log_on_failure = HOST
log_on_success = PID HOST EXIT
```

xinetd Network Security

For security restrictions, you can use **only_from** to restrict access by certain remote hosts. The **no_access** attribute denies access from the listed hosts, but no others. These controls take IP addresses as their values. You can list individual IP addresses, a range of IP addresses, or a network, using the network address. The **instances** attribute limits the number of server processes that can be active at once for a particular service. The following examples restrict access to a local network 192.168.1.0 and the localhost, deny access from 192.168.1.15, and use the **instances** attribute to limit the number of server processes at one time to 60.

```
only_from = 192.168.1.0
only_from = localhost
no_access = 192.168.1.15
instances = 60
```

The **xinetd** program also provides several internal services, including **time**, **services**, **servers**, and **xadmin**: **services** provides a list of currently active services, and **servers** provides information about servers; **xadmin** provides **xinetd** administrative support.

xinetd Service Configuration Files: /etc/xinetd.d Directory

Instead of having one large **xinetd.conf** file for all services, the service configurations are split it into several configuration files, one for each service. The directory is specified in **xinetd.conf** file with an **includedir** option. In the following example, the **xinetd.d** directory holds **xinetd** configuration files for services like SWAT. This approach has the advantage of letting you add services by just creating a new configuration file for them. Modifying a service involves editing only its configuration file, not an entire **xinetd.conf** file.

As an example, the **swat** file in the **xinetd.d** directory is shown here. Notice that it is disabled by default:

```
# default: off
# description: SWAT is the Samba Web Admin Tool.\
# Use swat to configure your Samba server. \
# To use SWAT, connect to port 901 with your \
# favorite web browser.
service swat
{
    port            = 901
    socket_type     = stream
    wait            = no
    only_from       = 127.0.0.1
    user            = root
    server          = /usr/sbin/swat
    log_on_failure  += USERID
    disable         yes
}
```

Configuring Services: xinetd Attributes

Entries in an **xinetd** service file define the server to be activated when requested along with any options and security precautions. An entry consists of a block of attributes defined for different features, such as the name of the server program, the protocol used, and security restrictions. Each block for an Internet service such as a server is preceded by the keyword **service** and the name by which you want to identify the service. A pair of braces encloses the block of attributes. Each attribute entry begins with the attribute name, followed by an assignment operator, such as **=**, and then the value or values assigned. A special block specified by the keyword **default** contains default attributes for services. The syntax is shown here:

```
service <service_name>
{
<attribute> <assign_op> <value> <value> ...
 ...
}
```

Most attributes take a single value for which you use the standard assignment operator, **=**. Some attributes can take a list of values. You can assign values with the **=** operator, but you can also add or remove items from these lists with the **+=** and **-=** operators. Use **+=** to add values and **-=** to remove values. You often use the **+=** and **-=** operators to add values to attributes that may have an initial value assigned in the default block.

Attributes are listed in Table 21-7. Certain attributes are required for a service. These include **socket_type** and **wait**. For a standard Internet service, you also need to provide the **user** (user ID for the service), the **server** (name of the server program), and the **protocol** (protocol used by the server). With **server_args**, you can also list any arguments you want passed to the server program (this does not include the server name). If **protocol** is not defined, the default protocol for the service is used.

Disabling and Enabling xinetd Services

You can turn services on or off manually by editing their **xinetd** configuration file. Services are turned on and off with the **disable** attribute in their configuration file. To enable a service, you set the disable attribute to **no**, as shown here:

```
disable = no
```

You then have to restart **xinetd** to start the service.

```
# /etc/init.d/xinetd restart
```

To enable management by **chkconfig**, a commented default and description entry needs to be placed before each service segment. Where separate files are used, the entry is placed at the head of each file. Many distributions already provide these for the services they install with their distributions, such as **tftp** and SWAT. A default entry can be either on or off. For example, the **chkconfig** default and description entries for the FTP service are shown here:

```
# default: off
# description: The tftp server serves files using the trivial file transfer \
#     protocol.  The tftp protocol is often used to boot diskless \
#     workstations, download configuration files to network-aware printers, \
#     and to start the installation process for some operating systems.
```

Attribute	Description
`ids`	Identifies a service. By default, the service ID is the same as the service name.
`type`	Type of service: `RPC`, `INTERNAL` (provided by **xinetd**), or `UNLISTED` (not listed in a standard system file).
`flags`	Possible flags include `REUSE`, `INTERCEPT`, `NORETRY`, `IDONLY`, `NAMEINARGS` (allows use of `tcpd`), `NODELAY`, and `DISABLE` (disable the service). See the **xinetd.conf** Man page for more details.
`disable`	Specify `yes` to disable the service.
`socket_type`	Specify `stream` for a stream-based service, `dgram` for a datagram-based service, `raw` for a service that requires direct access to IP, and `seqpacket` for reliable sequential datagram transmission.
`protocol`	Specifies a protocol for the service. The protocol must exist in **/etc/protocols**. If this attribute is not defined, the default protocol employed by the service will be used.
`wait`	Specifies whether the service is single-threaded or multithreaded (`yes` or `no`). If `yes`, the service is single-threaded, which means that **xinetd** will start the server and then stop handling requests for the service until the server stops. If `no`, the service is multithreaded and **xinetd** will continue to handle new requests for it.
`user`	Specifies the user ID (UID) for the server process. The username must exist in **/etc/passwd**.
`group`	Specifies the GID for the server process. The group name must exist in **/etc/group**.
`instances`	Specifies the number of server processes that can be simultaneously active for a service.
`nice`	Specifies the server priority.
`server`	Specifies the program to execute for this service.
`server_args`	Lists the arguments passed to the server. This does not include the server name.
`only_from`	Controls the remote hosts to which the particular service is available. Its value is a list of IP addresses. With no value, service is denied to all remote hosts.
`no_access`	Controls the remote hosts to which the particular service is unavailable.
`access_times`	Specifies the time intervals when the service is available. An interval has the form hour:min-hour:min.
`log_type`	Specifies where the output of the service log is sent, either the syslog facility (`SYSLOG`) or a file (`FILE`).
`log_on_success`	Specifies the information that is logged when a server starts and stops. Information you can specify includes `PID` (server process ID), `HOST` (the remote host address), `USERID` (the remote user), `EXIT` (exit status and termination signal), and `DURATION` (duration of a service session).

TABLE 21-7 Attributes for xinetd

Attribute	Description
`log_on_failure`	Specifies the information that is logged when a server cannot be started. Information you can specify includes **`HOST`** (the remote host address), **`USERID`** (user ID of the remote user), **`ATTEMPT`** (logs a failed attempt), and **`RECORD`** (records information from the remote host to allow monitoring of attempts to access the server).
`rpc_version`	Specifies the RPC version for an RPC service.
`rpc_number`	Specifies the number for an UNLISTED RPC service.
`env`	Defines environment variables for a service.
`passenv`	The list of environment variables from **xinetd**'s environment that will be passed to the server.
`port`	Specifies the service port.
`redirect`	Allows a TCP service to be redirected to another host.
`bind`	Allows a service to be bound to a specific interface on the machine.
`interface`	Synonym for **`bind`**.
`banner`	The name of a file to be displayed for a remote host when a connection to that service is established.
`banner_success`	The name of a file to be displayed at the remote host when a connection to that service is granted.
`banner_fail`	The name of a file to be displayed at the remote host when a connection to that service is denied.
`groups`	Allows access to groups the service has access to (**`yes`** or **`no`**).
`enabled`	Specifies the list of service names to enable.
`include`	Inserts the contents of a specified file as part of the configuration file.
`includedir`	Takes a directory name in the form of **`includedir /etc/xinetd.d`**. Every file inside that directory will be read sequentially as an **xinetd** configuration file, combining to form the **xinetd** configuration.

TABLE 21-7 Attributes for xinetd (*continued*)

If you want to turn on a service that is off by default, you can set its **`disable`** attribute to **`no`** and restart **xinetd**. The entry for the TFTP FTP server, **tftpd**, is shown here. An initial comment tells you that it is off by default, but then the **`disable`** attribute turns it on:

```
service tftp
{
  socket_type     = dgram
  protocol        = udp
  wait            = yes
  user            = root
  server          = /usr/sbin/in.tftpd
```

```
  server_args     = -s /tftpboot
  disable         = yes
  per_source      = 11
  cps             = 100 2
  flags           = IPv4
}
```

NOTE *You can also use* **xinetd** *to implement SSH port forwarding, should your system be used to tunnel connections between hosts or services.*

TCP Wrappers

TCP wrappers add another level of security to **xinetd**-managed servers. In effect, the server is wrapped with an intervening level of security, monitoring connections and controlling access. A server connection made through **xinetd** is monitored, verifying remote user identities and checking to make sure they are making valid requests. Connections are logged with the **syslogd** daemon (see Chapter 27) and may be found in **syslogd** files such as **/var/log/secure**. With TCP wrappers, you can also restrict access to your system by remote hosts. Lists of hosts are kept in the **hosts.allow** and **hosts.deny** files. Entries in these files have the format **service:hostname:domain**. The domain is optional. For the service, you can specify a particular service, such as FTP, or you can enter **ALL** for all services. For the hostname, you can specify a particular host or use a wildcard to match several hosts. For example, **ALL** will match on all hosts. Table 21-8 lists the available wildcards. In the following example, the first entry allows access by all hosts to the web service, **http**. The second entry allows access to all services by the **pango1.train.com** host. The third and fourth entries allow FTP access to **rabbit.trek.com** and **sparrow.com**:

```
http:ALL
ALL:pango1.train.com
ftp:rabbit.trek.com
ftp:sparrow.com
```

Wildcard	Description
ALL	Matches all hosts or services.
LOCAL	Matches any host specified with just a hostname without a domain name. Used to match on hosts in the local domain.
UNKNOWN	Matches any user or host whose name or address is unknown.
KNOWN	Matches any user or host whose name or address is known.
PARANOID	Matches any host whose hostname does not match its IP address.
EXCEPT	An operator that lets you provide exceptions to matches. It takes the form of *list1* **EXCEPT** *list2* where those hosts matched in *list1* that are also matched in *list2* are excluded.

TABLE 21-8 TCP Wrapper Wildcards

The **hosts.allow** file holds hosts to which you allow access. If you want to allow access to all but a few specific hosts, you can specify **ALL** for a service in the **hosts.allow** file but list the ones you are denying access to in the **hosts.deny** file. Using IP addresses instead of hostnames is more secure because hostnames can be compromised through the DNS records by spoofing attacks when an attacker pretends to be another host.

When **xinetd** receives a request for an FTP service, a TCP wrapper monitors the connection and starts up the **in.ftpd** server program. By default, all requests are allowed. To allow all requests specifically for the FTP service, you enter the following in your **/etc/hosts.allow** file. The entry **ALL:ALL** opens your system to all hosts for all services:

```
ftp:ALL
```

TIP *Originally, TCP wrappers were managed by the* ***tcpd*** *daemon. However,* ***xinetd*** *has since integrated support for TCP wrappers into its own program. You can explicitly invoke the* ***tcpd*** *daemon to handle services if you wish. The* ***tcpd*** *Man pages (**man tcpd**) provide more detailed information about* ***tcpd****.*

22 CHAPTER

FTP Servers

The File Transfer Protocol (FTP) is designed to transfer large files across a network from one system to another. Like most Internet operations, FTP works on a client/server model. FTP client programs can enable users to transfer files to and from a remote system running an FTP server program. Any Linux system can operate as an FTP server. It has to run only the server software—an FTP daemon with the appropriate configuration. Transfers are made between user accounts on client and server systems. A user on the remote system has to log in to an account on a server and can then transfer files to and from that account's directories only. A special kind of user account, named *ftp*, allows any user to log in to it with the username "anonymous." This account has its own set of directories and files that are considered public, available to anyone on the network who wants to download them. The numerous FTP sites on the Internet are FTP servers supporting FTP user accounts with anonymous login. Any Linux system can be configured to support anonymous FTP access, turning them into network FTP sites. Such sites can work on an intranet or on the Internet.

FTP Servers

FTP server software consists of an FTP daemon and configuration files. The *daemon* is a program that continuously checks for FTP requests from remote users. When a request is received, it manages a login, sets up the connection to the requested user account, and executes any FTP commands the remote user sends. For anonymous FTP access, the FTP daemon allows the remote user to log in to the FTP account using **anonymous** or **ftp** as the username. The user then has access to the directories and files set up for the FTP account. As a further security measure, however, the daemon changes the root directory for that session to be the FTP home directory. This hides the rest of the system from the remote user. Normally, any user on a system can move around to any directories open to him or her. A user logging in with anonymous FTP can see only the FTP home directory and its subdirectories. The remainder of the system is hidden from that user. This effect is achieved by the `chroot` operation (discussed later) that literally changes the system root directory for that user to that of the FTP directory. By default, the FTP server also requires a user be using a valid shell. It checks for a list of valid shells in the **/etc/shells** file. Most daemons have options for turning off this feature.

Available Servers

Several FTP servers are available for use on Linux systems (see Table 22-1). Three of the more common servers include **vsftpd**, **pureftpd**, and **proftpd**. The Very Secure FTP Server provides a simple and very secure FTP server. The Pure FTPD server is a lightweight, fast, and secure FTP server, based upon Troll-FTPd. Documentation and the latest sources are available from **pureftpd.org**.

ProFTPD is a popular FTP daemon based on an Apache web server design. It features simplified configuration and support for virtual FTP hosts. The compressed archive of the most up-to-date version, along with documentation, is available at the ProFTPD website at **proftpd.org**. Another FTP daemon, NcFTPd, is a commercial product produced by the same programmers who did the NcFTP FTP client. NcFTPd is free for academic use and features a reduced fee for small networks. Check **ncftpd.org** for more information.

Several security-based FTP servers are also available, including SSLFTP and SSH **sftpd**, along with **gssftpd**. SSLFTP uses SSL (Secure Sockets Layer) to encrypt and authenticate transmissions, as well as MD5 digests to check the integrity of transmitted files. SSH **sftpd** is an FTP server that is now part of the Open SSH package, using SSH encryption and authentication to establish secure FTP connections. The **gssftpd** server is part of the Kerberos 5 package and provides Kerberos-level security for FTP operations.

FTP Users

Normal users with accounts on an FTP server can gain full FTP access simply by logging into their accounts. Such users can access and transfer files directly from their own accounts or any directories they may have access to. You can also create users, known as guest users, that have restricted access to the FTP publicly accessible directories. This involves setting standard user restrictions, with the FTP public directory as their home directory. Users can also log in as anonymous users, allowing anyone on the network or Internet to access files on an FTP server.

FTP Server	Site
Very Secure FTP Server (vsftpd)	**vsftpd.beasts.org**
ProFTPD	**proftpd.org**
PureFTP	**pureftpd.org**
NcFTPd	**ncftpd.org**
SSH sftp	**openssh.org**
Washington University web server (WU-FTPD)	**wu-ftpd.org**
Tux	Web server with FTP capabilities
gssftpd	Kerberos FTP server

Table 22-1 FTP Servers

Anonymous FTP: vsftpd

An anonymous FTP site is essentially a special kind of user on your system with publicly accessible directories and files in its home directory. Anyone can log in to this account and access its files. Because anyone can log in to an anonymous FTP account, you must be careful to restrict a remote FTP user to only the files on that anonymous FTP directory. Normally, a user's files are interconnected to the entire file structure of your system. Normal users have write access that lets them create or delete files and directories. The anonymous FTP files and directories can be configured in such a way that the rest of the file system is hidden from them and remote users are given only read access. In ProFTPD, this is achieved through configuration directives placed in its configuration file. An older approach implemented by the vsftpd package involves having copies of certain system configuration, command, and library files placed within subdirectories of the FTP home directory. Restrictions placed on those subdirectories then control access by other users. Within the FTP home directory, you then have a publicly accessible directory that holds the files you want to make available to remote users. This directory usually has the name **pub**, for public.

An FTP site is made up of an FTP user account, an FTP home directory, and certain copies of system directories containing selected configuration and support files. Newer FTP daemons, such as ProFTPD, do not need the system directories and support files. Most distributions have already set up an FTP user account when you installed your system.

The FTP User Account: anonymous

To allow anonymous FTP access by other users to your system, you must have a user account named *FTP*. Most distributions already create this account for you. If your system does not have such an account, you will have to create one. You can then place restrictions on the FTP account to keep any remote FTP users from accessing any other part of your system. You must also modify the entry for this account in your **/etc/passwd** file to prevent normal user access to it. The following is the entry you find in your **/etc/passwd** file that sets up an FTP login as an anonymous user:

```
ftp:x:14:50:FTP User:/var/ftp:
```

The **x** in the password field blocks the account, which prevents any other users from gaining access to it, thereby gaining control over its files or access to other parts of your system. The user ID, 14, is a unique ID. The comment field is FTP User. The login directory is **/var/ftp**. When FTP users log in to your system, they are placed in this directory. If a home directory has not been set up, create one and then change its ownership to the FTP user with the **chown** command.

FTP Group

The group ID is the ID of the **ftp** group, which is set up only for anonymous FTP users. You can set up restrictions on the **ftp** group, thereby restricting any anonymous FTP users. Here is the entry for the **ftp** group you find in the **/etc/group** file. If your system does not have one, you should add it:

```
ftp::50:
```

Creating New FTP Users

If you are creating virtual FTP hosts, you will need to create an FTP user for each one, along with its directories. For example, to create an FTP server for a host1-ftp host, you create a host1-ftp user with its own directory.

```
useradd -d /var/host1-ftp host1-ftp
```

This creates a user such as that described here:

```
host1-ftp:x:14:50:FTP User:/var/host1-ftp:
```

You also need to create the corresponding home directory, **/var/host1-ftp** in this example, and set its permissions to give users restricted access.

```
mkdir /var/host1-ftp
chmod 755 /var/host1-ftp
```

In addition, you need to make sure that the root user owns the directory, not the new FTP users. This gives control of the directory only to the root user, not to any user that logs in.

```
chown root.root /var/host1-ftp
```

Anonymous FTP Server Directories

As previously noted, the FTP home directory is named **ftp** and is placed in the **/var** directory. When users log in anonymously, they are placed in this directory. An important part of protecting your system is preventing remote users from using any commands or programs not in the restricted directories. For example, you would not let a user use your **ls** command to list filenames, because **ls** is located in your **/bin** directory. At the same time, you want to let the FTP user list filenames using an **ls** command. Newer FTP daemons such as vsftpd and ProFTPD solve this problem by creating secure access to needed system commands and files, while restricting remote users to only the FTP site's directories. In any event, make sure that the FTP home directory is owned by the root user, not by the FTP user. Use the **ls -d** command to check on the ownership of the FTP directory.

```
ls -d /var/ftp
```

To change a directory's ownership, you use the **chown** command, as shown in this example:

```
chown  root.root /var/ftp
```

Another, more traditional solution is to create copies of certain system directories and files needed by remote users and to place them in the **ftp** directory where users can access them. A **bin** directory is placed in the **ftp** directory and remote users are restricted to it, instead of the system **bin** directory. Whenever they use the **ls** command, remote users are using the one in **ftp/bin**, not the one you use in **/bin**. If, for some reason, you set up the anonymous FTP directories yourself, you must use the **chmod** command to change the access permissions for the directories so that remote users cannot access the rest of your system. Create an **ftp** directory and use the **chmod** command with the permission 555 to

turn off write access: **chmod 555 ftp**. Next, make a new **bin** directory in the **ftp** directory, and then make a copy of the **ls** command and place it in **ftp/bin**. Do this for any commands you want to make available to FTP users. Then create an **ftp/etc** directory to hold a copy of your **passwd** and **group** files. Again, the idea is to prevent any access to the original files in the **/etc** directory by FTP users. The **ftp/etc/passwd** file should be edited to remove any entries for regular users on your system. All other entries should have their passwords set to **x** to block access. For the **group** file, remove all user groups and set all passwords to **x**. Create an **ftp/lib** directory, and then make copies of the libraries you need to run the commands you placed in the **bin** directory.

Anonymous FTP Files

A directory named **pub**, located in the FTP home directory, usually holds the files you are making available for downloading by remote FTP users. When FTP users log in, they are placed in the FTP home directory (**/var/ftp**), and they can then change to the **pub** directory to start accessing those files (**/var/ftp/pub**). Within the **pub** directory, you can add as many files and directories as you want. You can even designate some directories as upload directories, enabling FTP users to transfer files to your system.

In each subdirectory set up under the **pub** directory to hold FTP files, you should create a **README** file and an **INDEX** file as a courtesy to FTP users. The **README** file contains a brief description of the kind of files held in this directory. The **INDEX** file contains a listing of the files and a description of what each one holds.

Using FTP with rsync

Many FTP servers also support rsync operations using **rsync** as a daemon. This allows intelligent incremental updates of files from an FTP server. You can update multiple files in a directory or a single file such as a large ISO image.

Accessing FTP Sites with rsync

To access the FTP server running an rsync server, you enter the **rsync** command, and following the hostname, you enter a double colon and then either the path of the directory you want to access or one of the FTP server's modules. In the following example, the user updates a local **myproject** directory from the one on the **mytrek.com** FTP site:

```
rsync ftp.mytrek.com::/var/ftp/pub/myproject  /home/myproject
```

To find out what directories are supported by rsync, you check for rsync modules on that site. These are defined by the site's **/etc/rsyncd.conf** configuration file. A *module* is just a directory with all its subdirectories. To find available modules, you enter the FTP site with a double colon only.

```
rsync ftp.mytrek.com::
ftp
```

This tells you that the **ftp.mytrek.com** site has an FTP module. To list the files and directories in the module, you can use the **rsync** command with the **-r** option.

```
rsync -r ftp.mytrek.com::ftp
```

Many sites that run the rsync server will have an rsync protocol that will already be set to access the available rsync module (directory). For example, the following URL can be used with **rsync** to access the ibiblio location for your distribution. For Fedora, the module is named fedora-linux-core, which follows the hostname.

```
rsync://distro.ibiblio.org/fedora-linux-core/
```

You can even use **rsync** to update just a single file, such as an ISO image that may have been changed. The following example updates the Fedora 7 ISO disk 1 image. The **--progress** option will show the download progress.

```
rsync -a --progress rsync://distro.ibiblio.org/fedora-linux-core/7/i386/
iso/FC7-i386-disc1.iso
```

Configuring an rsync Server

To configure your FTP server to let clients use rsync on your site, you need to first run rsync as server. Use **chkconfig** or **sysv-rc-conf** to turn on the rsync daemon, commonly known as **rsyncd**. This will run the rsync daemon through **xinetd**, using an **rsync** script in **/etc/xinetd.d** to turn it on and set parameters.

```
chkconfig rsync on
```

When run as a daemon, rsync will read the **/etc/rsyncd.conf** file for its configuration options. Here you can specify FTP options such as the location for the FTP site files. The configuration file is segmented into modules, each with its own options. A module is a symbolic representation of an exported tree (a directory and its subdirectories). The module name is enclosed in brackets, for instance, **[ftp]** for an FTP module. You can then enter options for that module, as by using the path option to specify the location of your FTP site directories and files. The user and group IDs can be specified with the **uid** and **gid** options. The default is nobody. A sample FTP module for anonymous access is shown here:

```
[ftp]
       path = /var/ftp/pub
       comment = ftp site
```

For more restricted access, you can add an **auth users** option to specify authorized users; rsync will allow anonymous access to all users by default. Access to areas on the FTP site by rsync can be further controlled using a secrets file, such as **/etc/rsyncd.secrets**. This is a colon-separated list of usernames and passwords.

```
aleina:mypass3
larisa:yourp5
```

A corresponding module to the controlled area would look like this:

```
[specialftp]
      path = /var/projects/special
      command = restricted access
      auth users = aleina,larisa
      secrets file = /etc/rsyncd.secrets
```

If you are on your FTP server and want to see what modules will be made available, you can run **rsync** with the **localhost** option and nothing following the double colon.

```
$ rsync localhost::
ftp
specialftp
```

Remote users can find out what modules you have by entering your hostname and double colon only.

```
rsync ftp.mytrek.com::
```

rsync Mirroring

Some sites will allow you to use rsync to perform mirroring operations. With rsync you do not have to copy the entire site, just those files that have been changed. The following example mirrors the mytrek FTP site to the **/var/ftp/mirror/mytrek** directory on a local system:

```
rsync -a --delete ftp.mytrek.com::ftp /var/ftp/mirror/mytrek
```

The **-a** option is archive mode, which includes several other options, such as **-r** (recursive) to include all subdirectories, **-t** to preserves file times and dates, **-l** to recreate symbolic links, and **-p** to preserve all permissions. In addition, the **--delete** option is added to delete files that don't exist on the sending side, removing obsolete files.

The Very Secure FTP Server

The Very Secure FTP Server (vsftpd) is small, fast, easy, and secure. It is designed to avoid the overhead of large FTP server applications like ProFTPD, while maintaining a very high level of security. It can also handle a very large workload, managing high traffic levels on an FTP site. It is perhaps best for sites where many anonymous and guest users will be downloading the same files. It replaced the Washington University FTP server, WU-FTPD, on many distributions.

The Very Secure FTP Server is inherently designed to provide as much security as possible, taking full advantage of Unix and Linux operating system features. The server is separated into privileged and unprivileged processes. The unprivileged process receives all FTP requests, interpreting them and then sending them over a socket to the privileged process, which then securely filters all requests. Even the privileged process does not run with full root capabilities, using only those that are necessary to perform its tasks. In addition, the Very Secure FTP Server uses its own version of directory commands like **ls**, instead of the system's versions.

Running vsftpd

The Very Secure FTP Server's daemon is named **vsftpd**. It is designed to be run as a standalone server, which can be started and stopped using the **/etc/rc.d/init.d/vsftpd** server script. To have the server start automatically, you can turn it on with the **chkconfig** command and the **on** argument, GNOME's services admin, rrconf, or sysv-rc-conf to turn

the vsftpd service on or off. Turn off the service to disable the server. If you previously enabled another FTP server such as ProFTPD, be sure to disable it first.

Alternatively, you can implement **vsftpd** to be run by **xinetd**, running the server only when a request is made by a user. The **xinetd** daemon will run an **xinetd** script file called **vsftpd,** located in the **/etc/xinetd.d** directory. Initially, the server will be turned off.

Configuring vsftpd

You configure **vsftpd** using one configuration file, **vsftpd.conf**. Configuration options are simple and kept to a minimum, making it less flexible than ProFTPD, but much faster (see Table 22-2). The **vsftpd.conf** file contains a set of directives in which an option is assigned a value (there are no spaces around the = sign). Options can be on and off flags assigned a **YES** or **NO** value, features that take a numeric value, or ones that are assigned a string. A default **vsftpd.conf** file is installed in the **/etc or /etc reference/etc/vsftpd** directory. This file lists some of the commonly used options available, with detailed explanations for each. Those that are not used are commented out with a preceding # character. Option names are very understandable. For example, **anon_upload_enable** allows anonymous users to upload files, whereas **anon_mkdir_write_enable** lets anonymous users create directories. The Man page for **vsftpd.conf** lists all options, providing a detailed explanation for each.

Enabling Standalone Access

To run **vsftpd** as a standalone server, you set the listen option to **YES**. This instructs **vsftpd** to continually listen on its assigned port for requests. You can specify the port it listens on with the **listen_port** option.

```
listen=YES
```

Enabling Login Access

In the following example taken from the **vsftpd.conf** file, anonymous FTP is enabled by assigning the **YES** value to the **anonymous_enable** option. The **local_enable** option allows local users on your system to use the FTP server.

```
# Allow anonymous FTP?
anonymous_enable=YES
#
# Uncomment this to allow local users to log in.
local_enable=YES
```

Should you want to let anonymous users log in without providing a password, you can set **no_anon_password** to YES.

Local User Permissions

A variety of user permissions control how local users can access files on the server. If you want to allow local users to create, rename, and delete files and directories on their account, you have to enable write access with the **write_enable** option. This way, any files they upload, they can also delete. Literally, the **write_enable** option activates a range of commands for changing the file system, including creating, renaming, and deleting both files and directories. With user_config_dir you can configure specific users.

```
write_enable=YES
```

Option	Description
`listen`	Set standalone mode.
`listen_port`	Specify port for standalone mode.
`anonymous_enable`	Enable anonymous user access.
`local_enable`	Enable access by local users.
`no_anon_password`	Specify whether anonymous users must submit a password.
`anon_upload_enable`	Enable uploading by anonymous users.
`anon_mkdir_write_enable`	Allow anonymous users to create directories.
`anon_world_readable_only`	Make uploaded files read-only to all users.
`idle_session_timeout`	Set time limit in seconds for idle sessions.
`data_connection_timeouts`	Set time limit in seconds for failed connections.
`dirmessage_enable`	Display directory messages.
`ftpd_banner`	Display FTP login message.
`xferlog_enable`	Enable logging of transmission transactions.
`xferlog_file`	Specify log file.
`deny_email_enable`	Enable denying anonymous users, whose email addresses are specified in **vsftpd.banned**.
`userlist_enable`	Deny access to users specified in the **vsftp.user_list** file.
`userlist_file`	Deny or allow users access depending on setting of `userlist_deny`.
`userlist_deny`	When set to `YES`, **userlist_file** denies list users access. When set to `NO`, **userlist_file** allows list users, and only those users, access.
`chroot_list_enable`	Restrict users to their home directories.
`chroot_list_file`	Allow users access to home directories. Unless `chroot_local_user` is set to `YES`, this file contains a list of users not allowed access to their home directories.
`chroot_local_user`	Allow access by all users to their home directories.
`user_config_dir`	Directory for user specific configurability
`ls_recurse_enable`	Enable recursive listing.

TABLE 22-2 Configuration Options for vsftpd.conf

You can further specify the permissions for uploaded files using the `local_umask` option (022 is the default set in **vsftpd.conf**, which allows read and write for the owner and read-only for all other users, 644).

```
local_umask=022
```

Because ASCII uploads entail certain security risks, they are turned off by default. However, if you are uploading large text files, you may want to enable them in special cases. Use **ascii_upload_enable** to allow ASCII uploads.

Anonymous User Permissions

You can also allow anonymous users to upload and delete files, as well as create or remove directories. Uploading by anonymous users is enabled with the **anon_upload_enable** option. To let anonymous users also rename or delete their files, you set the **anon_other_write_enable** option. To let them create directories, you set the **anon_mkdir_write_enable** option.

```
anon_upload_enable=YES
anon_other_write_enable=YES
anon_mkdir_write_enable=YES
```

The **anon_world_readable_only** option will make uploaded files read-only (downloadable), restricting write access to the user that created them. Only the user that uploaded a file can delete it.

All uploaded files are owned by the anonymous FTP user. You can have the files owned by another user, adding greater possible security. In effect, the actual user owning the uploaded files becomes hidden from anonymous users. To enable this option, you use **chown_uploads** and specify the new user with **chown_username**. Never make the user an administrative user like **root**.

```
chown_uploads=YES
chown_username=myftpfiles
```

The upload directory itself should be given write permission by other users.

```
chmod 777 /var/ftp/upload
```

You can control the kind of access that users have to files with the **anon_mask** option, setting default read/write permissions for uploaded files. The default is 077, which gives read/write permission to the owner only (600). To allow all users read access, you set the umask to 022, where the 2 turns off write permission but sets read permission (644). The value 000 allows both read and write for all users.

Connection Time Limits

To more efficiently control the workload on a server, you can set time limits on idle users and failed transmissions. The **idle_session_timeout** option will cut off idle users after a specified time, and **data_connection_timeouts** will cut off failed data connections. The defaults are shown here:

```
idle_session_timeout=600
data_connection_timeout=120
```

Messages

The **dirmessage_enable** option allows a message held in a directory's **.message** file to be displayed whenever a user accesses that directory. The **ftpd_banner** option lets you set up your own FTP login message. The default is shown here:

```
ftpd_banner=Welcome to blah FTP service.
```

Logging

A set of **xferlog** options control logging. You can enable logging, as well as specify the format and the location of the file.

```
xferlog_enable=YES
```

Use the **xferlog_file** option to specify the log file you want to use. The default is shown here:

```
xferlog_file=/var/log/vsftpd.log
```

vsftpd Access Controls

Certain options control access to the FTP site. As previously noted, the **anonymous_enable** option allows anonymous users access, and **local_enable** permits local users to log in to their accounts. (If /etc/vsftpd exits, no vsftpd. prefix is used on files).

Denying Access

The **deny_email_enable** option lets you deny access by anonymous users, and the **banned_email** file option designates the file (usually **vsftpd.banned**) that holds the email addresses of those users. The **vsftpd.ftpusers** file lists those users that can never access the FTP site. These are usually system users like **root**, **mail**, and **nobody**. See Table 22-3 for a list of vsftpd files.

User Access

The **userlist_enable** option controls access by users, denying access to those listed in the file designated by the **userlist_file** option (usually **vsftpd.user_list**). If, instead, you want to restrict access to just certain select users, you can change the meaning and usage of the **vsftpd.user_list** file to indicate only those users allowed access, instead of those denied access. To do this, you set the **userlist_deny** option to **NO** (its default is **YES**). Only users listed in the **vsftpd.user_list** file will be granted access to the FTP site.

File	Description
vsftpd.ftpusers	List of users always denied access
vsftpd.user_list	Specified users denied access (allowed access if **userlist_deny** is **NO**)
vsftpd.chroot_list	List of local users allowed access (denied access if **chroot_local_user** is on)
/etc/vsftpd/vsftpd.conf	vsftpd configuration file (or /etc/vsftpd/vsftpd.conf)
/etc/pam.d/vsftpd	PAM vsftpd script
/etc/rc.d/init.d/vsftpd	Service vsftpd server script, standalone
/etc/xinetd.d/vsftpd	Xinetd vsftpd server script

Table 22-3 Files for vsftpd

User Restrictions

The **chroot_list_enable** option controls access by local users, letting them access only their home directories, while restricting system access. The **chroot_list_file** option designates the file (usually **vstfpd.chroot**) that lists those users allowed access. You can allow access by all local users with the **chroot_local_user** option. If this option is set, then the file designated by **chroot_list_file** will have an inverse meaning, listing those users not allowed access. In the following example, access by local users is limited to those listed in **vsftpd.chroot**:

```
chroot_list_enable=YES
chroot_list_file=/etc/vsftpd.chroot_list
```

User Authentication

The **vsftpd** server makes use of the PAM service to authenticate local users that are remotely accessing their accounts through FTP. In the **vsftpd.conf** file, the PAM script used for the server is specified with the **pam_service_name** option.

```
pam_service_name=vsftpd
```

In the **etc/pam.d** directory, you will find a PAM file named **vsftpd** with entries for controlling access to the **vsftpd** server. PAM is currently set up to authenticate users with valid accounts, as well as deny access to users in the **/etc/vsftpd.ftpusers** file. The default **/etc/pam.d/vsftpd** file is shown here:

```
#%PAM-1.0
auth required pam_listfile.so item=user sense=deny
                 file=/etc/vsftpd.ftpusers onerr=succeed
auth    required  pam_stack.so service=system-auth
auth   required   pam_shells.so
account required  pam_stack.so service=system-auth
session required  pam_stack.so service=system-auth
```

Command Access

Command usage is highly restricted by vsftpd. Most options for the **ls** command that lists files are not allowed. Only the asterisk file-matching operation is supported. To enable recursive listing of files in subdirectories, you have to enable the use of the **-R** option by setting the **ls_recurse_enable** option to **YES**. Some clients, such as **ncftp**, will assume that the recursive option is enabled.

vsftpd Virtual Hosts

Though the capability is not inherently built in to vsftpd, you can configure and set up the vsftpd server to support virtual hosts. *Virtual hosting* is where a single FTP server operates as if it has two or more IP addresses. Several IP addresses can then be used to access the same server. The server will then use a separate FTP user directory and files for each host. With vsftpd, this involves manually creating separate FTP users and directories for each virtual host, along with separate vsftpd configuration files for each virtual host in the **/etc/vsftpd** directory. **vsftpd** is configured to run as a standalone service. Its **/etc/rc.d/init.d/vsftpd** startup script will automatically search for and read any configuration files listed in the **/etc/vsftpd** directory.

If, on the other hand, you wish to run **vsftpd** as an **xinetd** service, you have to create a separate **xinetd** service script for each host in the **/etc/xinetd.d** directory. In effect, you have several **vsftpd** services running in parallel for each separate virtual host. The following example uses two IP addresses for an FTP server:

1. Create an FTP user for each host. Create directories for each host (you can use the one already set up for one of the users). For example, for the first virtual host you could use **FTP-host1**. Be sure to set root ownership and the appropriate permissions.

```
useradd -d /var/ftp-host1 FTP-host1
chown root.root /var/ftp-host1
chmod a+rx /var/ftp-host1
umask 022
mkdir /var/ftp-host1/pub
```

2. Set up two corresponding vsftpd service scripts in the **/etc/xinetd.d** directory. The **vsftpd** directory in **/usr/share/doc** has an **xinetd** example script, **vsftpd.xinetd**. Within each, enter a **bind** command to specify the IP address the server will respond to.

```
bind  192.168.0.34
```

3. Within the same scripts, enter a **server_args** entry specifying the name of the configuration file to use.

```
server_args = vsftpd-host1.conf
```

4. Within the **/etc/vsftpd** directory, create separate configuration files for each virtual host. Within each, specify the FTP user you created for each, using the **ftp_username** entry.

```
ftp_username = FTP-host1
```

vsftpd Virtual Users

Virtual users can be implemented by making use of PAM to authenticate authorized users. In effect, you are allowing access to certain users, while not having to actually set up accounts for them on the FTP server system. First, create a PAM login database file to use along with a PAM file in the **/etc/pam.d** directory that will access the database. Then create a virtual FTP user along with corresponding directories that the virtual users will access (see the vsftpd documentation at **vsftpd.beasts.org** for more detailed information). Then, in the **vsftpd.conf** file, you can disable anonymous FTP:

```
anonymous_enable=NO
local_enable=YES
```

and then enable guest access:

```
guest_enable=YES
guest_username=virtual
```

Professional FTP Daemon: ProFTPD

ProFTPD is based on the same design as the Apache web server, implementing a similar simplified configuration structure and supporting such flexible features as virtual hosting. ProFTPD is an open source project made available under a GNU GPL. You can download the current version from its website at **proftpd.org**. There you will also find detailed documentation including FAQs, user manuals, and sample configurations. Check the site for new releases and updates.

ProFTPD is designed to be extensible, supporting dynamic modules that can be either custom made or provided by third-party developers. Currently, there are modules to provide LDAP and SQL authentication, along with numerous enhancement modules for such features as user quotas, logging formats, and time constraints.

Install and Startup

If you install ProFTPD using distribution packages, the required configuration entries are made in your **proftpd.conf** file. If you install from compiled source code, you may have to modify the entries in the default **proftpd.conf** file provided. Make sure the FTP user and group specified in the **proftpd.conf** file actually exist.

To set the runlevels at which it will start automatically, you can use **`chkconfig`** or sysv-rc-conf. The following command sets **proftpd** to run automatically from runlevels 3 and 5:

```
chkconfig --level 35 proftpd on
```

To turn the service on for default levels, you can use **rcconf** or **services-admin**.

Authentication

Authentication of users in ProFTPD can be implemented either in PAM, LDAP, or SQL. For LDAP or SQL, you need to load the corresponding ProFTPD module. PAM authentication is the default (see Chapter 30 for more information on LDAP and PAM). Be sure that an **/etc/pam.d/ftp** file has been installed to provide PAM FTP support. ProFTPD also supports both standard passwords and shadow passwords. The default PAM authentication can be turned off by setting the **AuthPAMAuthoritative** directive to off, allowing other authentication methods such as LDAP to be used. If your server is further hidden in a local network by IP masquerading techniques, you can use the **MasqueradeAddress** directive to specify its public hostname, allowing its real hostname to remain hidden.

proftpd.config and .ftpaccess

ProFTPD uses only one configuration file, named **proftpd.conf**, located in the **/etc** directory. Configuration entries take the form of directives. This format is purposely modeled on Apache configuration directives. With the directives, you can enter basic configuration information, such as your server name, or perform more complex operations, such as implementing virtual FTP hosts. The design is flexible enough to enable you to define configuration features for particular directories, users, or groups.

To configure a particular directory, you can use an **.ftpaccess** file with configuration options placed within that directory. These **.ftpaccess** options take precedence over those in the **proftpd.conf** directory. The **.ftpaccess** files are designed to operate like **.htaccess** files in

the Apache web server that configure particular website directories. You can find a complete listing of ProFTPD configuration parameters at the ProFTPD website (**proftpd.org**) and in the ProFTPD documentation installed in **/usr/doc** as part of the ProFTPD software package. When creating a new configuration, you should make a copy of the **proftpd.conf** configuration file and modify it. Then you can test its syntax using the **proftpd** command with the **-c** option and the name of the file.

```
proftpd -c newfile.conf
```

Different kinds of directives exist. Many set values, such as **MaxClients**, which sets the maximum number of clients, or **NameServer**, which sets the name of the FTP server. Others create blocks that can hold directives that apply to specific FTP server components. Block directives are entered in pairs: a beginning directive and a terminating directive. The terminating directive defines the end of the block and consists of the same name, beginning with a slash. Block directives take an argument that specifies the particular object to which the directives will apply. For the **Directory** block directive, you must specify a directory name to which it will apply. The **<Directory** *mydir***>** block directive creates a block whose directives within it apply to the *mydir* directory. The block is terminated by a **</Directory>** directive. **<Anonymous** *ftp-dir***>** configures the anonymous service for your FTP server. You need to specify the directory on your system used for your anonymous FTP service, such as **/var/ftp**. The block is terminated with the **</Anonymous>** directive. The **<VirtualHost** *hostaddress***>** block directive is used to configure a specific virtual FTP server and must include the IP or the domain name address used for that server. **</VirtualHost>** is its terminating directive. Any directives you place within this block are applied to that virtual FTP server. The **<Limit** *permission***>** directive specifies the kind of access you want to limit. It takes as its argument one of several keywords indicating the kind of permission to be controlled: **WRITE** for write access, **READ** for read access, **STOR** for transfer access (uploading), and **LOGIN** to control user login.

A sample of the standard **proftpd.conf** file installed as part of the ProFTPD software package is shown here. It establishes a single server and a single anonymous login. Notice the default **ServerType** is **standalone**. If you want to use **xinetd** to run your server, you must change this entry to **inetd**. Detailed examples of **proftpd.conf** files, showing various anonymous FTP and virtual host configurations, can be found with the ProFTPD documentation, located in **/usr/share/doc**, and on the ProFTPD website at **proftpd.org**.

```
ServerName          "ProFTPD Default Installation"
ServerType          standalone
DefaultServer       on
Port             21
Umask            022
MaxInstances     30
User           nobody
Group          nobody
<Directory /*>
       AllowOverwrite      on
</Directory>
# A basic anonymous configuration, with one incoming directory.
<Anonymous ~ftp>
  User             ftp
```

```
  Group           ftp
  RequireValidShell      off
  MaxClients         10
  UserAlias         anonymous ftp
  DisplayLogin         welcome.msg
  DisplayFirstChdir      .message
  # Limit WRITE everywhere in the anonymous chroot except incoming
  <Directory *>
     <Limit WRITE>
         DenyAll
     </Limit>
  </Directory>
  <Directory incoming>
     <Limit WRITE>
            AllowAll
     </Limit>
     <Limit READ>
            DenyAll
     </Limit>
  </Directory>
</Anonymous>
```

Anonymous Access

You use the **Anonymous** configuration directive to create an anonymous configuration block in which you can place directives that configure your anonymous FTP service. The directive includes the directory on your system used for the anonymous FTP service. The ProFTPD daemon executes a **`chroot`** operation on this directory, making it the root directory for the remote user accessing the service. By default, anonymous logins are supported, expecting users to enter their email address as a password. You can modify an anonymous configuration to construct more controlled anonymous services, such as guest logins and required passwords.

NOTE *For ProFTPD, your anonymous FTP directory does not require any system files. Before ProFTPD executes a* **`chroot`** *operation, hiding the rest of the system from the directory, it accesses and keeps open any needed system files outside the directory.*

The following example shows a standard anonymous FTP configuration. The initial **Anonymous** directive specifies **/var/ftp** as the anonymous FTP home directory. The **User** directive specifies the user that the Anonymous FTP daemon will run as, and **Group** indicates its group. In both cases, FTP, the standard username, is used on most systems for anonymous FTP. A **Directory** directive with the **`*`** file-matching character then defines a Directory block that applies to all directories and files in **/var/ftp**. The **`*`** symbol matches on all filenames and directories. Within the **Directory** directive is a **Limit** directive that you use to place controls on a directory. The directive takes several arguments, including **`READ`** for read access and **`WRITE`** for write access. In this example, the **Limit** directive places restrictions on the write capabilities of users. Within the **Limit** directive, the **DenyAll** directive denies write permission, preventing users from creating or deleting files and effectively giving them only read access. A second **Directory** directive creates an exception to this rule for the incoming directory. An incoming directory is usually set up on FTP sites

to let users upload files. For this directory, the first **Limit** directive prevents both **READ** and **WRITE** access by users with its **DenyAll** directive, effectively preventing users from deleting or reading files here. The second **Limit** directive allows users to upload files, however, permitting transfers only (**STOR**) with the **AllowAll** directive.

One important directive for anonymous FTP configurations is **RequireValidShell**. By default, the FTP daemon first checks to see if the remote user is attempting to log in using a valid shell, such as the BASH shell or the C shell. The FTP daemon obtains the list of valid shells from the **/etc/shells** file. If the remote user does not have a valid shell, a connection is denied. You can turn off the check using the **RequireValidShell** directive and the **off** option. The remote user can then log in using any kind of shell.

```
<Anonymous /var/ftp>
    User ftp
    Group ftp
    UserAlias anonymous ftp
    RequireValidShell off
 <Directory *>
       <Limit WRITE>
               DenyAll
       </Limit>
    </Directory>
     # The only command allowed in incoming is STOR
     # (transfer file from client to server)
    <Directory incoming>
       <Limit READ WRITE>
               DenyAll
       </Limit>
       <Limit STOR>
               AllowAll
       </Limit>
    </Directory>
 </Anonymous>
```

Recall that FTP was originally designed to let a remote user connect to an account of his or her own on the system. Users can log in to different accounts on your system using the FTP service. Anonymous users are restricted to the anonymous user account. However, you can create other users and their home directories that also function as anonymous FTP accounts with the same restrictions. Such accounts are known as *guest accounts*. Remote users are required to know the username and, usually, the password. Once connected, they have only read access to that account's files; the rest of the file system is hidden from them. In effect, you are creating a separate anonymous FTP site at the same location with more restricted access.

To create a guest account, first create a user and the home directory for it. You then create an Anonymous block in the **proftpd.conf** file for that account. The **Anonymous** directive includes the home directory of the guest user you create. You can specify this directory with a **~** for the path and the directory name, usually the same as the username. Within the Anonymous block, you use the **User** and **Group** directives to specify the user and group name for the user account. Set the **AnonRequirePassword** directive to **on** if you want remote users to provide a password. A **UserAlias** directive defines aliases for the username. A remote user can use either the alias or the original username to log in. You then

enter the remaining directives for controlling access to the files and directories in the account's home directory. An example showing the initial directives is listed here. The **User** directive specifies the user as **myproject**. The home directory is **~myproject**, which usually evaluates to **/var/myproject**. The **UserAlias** directive lets remote users log in with either the name **myproject** or **mydesert**.

```
<Anonymous ~myproject>
    User myproject
    Group other
    UserAlias mydesert myproject
    AnonRequirePassword on
    <Directory *>
```

You could just as easily create an account that requires no password, letting users enter in their email address instead. The following example configures an anonymous user named **mypics**. A password isn't required, and neither is a valid shell. The remote user still needs to know the username, in this case **mypics**.

```
<Anonymous /var/mypics>
    AnonRequirePassword off
    User mypics
    Group nobody
    RequireValidShell off
    <Directory *>
```

The following example provides a more generic kind of guest login. The username is **guest**, with the home directory located at **~guest**. Remote users are required to know the password for the guest account. The first **Limit** directive lets all users log in. The second **Limit** directive allows write access from users on a specific network, as indicated by the network IP address, and denies write access by any others.

```
<Anonymous ~guest>
  User            guest
  Group           nobody
  AnonRequirePassword      on
  <Limit LOGIN>
         AllowAll
  </Limit>
  # Deny write access from all except trusted hosts.
  <Limit WRITE>
         Order        allow,deny
         Allow        from 10.0.0.
         Deny         from all
  </Limit>
</Anonymous>
```

Virtual FTP Servers

The ProFTPD daemon can manage more than one FTP site at once. Using a **VirtualHost** directive in the **proftpd.conf** file, you can create an independent set of directives that configure a separate FTP server. The **VirtualHost** directive is usually used to configure

virtual servers as FTP sites. You can configure your system to support more than one IP address. The extra IP addresses can be used for virtual servers, not independent machines. You can use such an extra IP address to set up a virtual FTP server, giving you another FTP site on the same system. This added server would use the extra IP address as its own. Remote users could access it using that IP address, instead of the system's main IP address. Because such an FTP server is not running independently on a separate machine but is, instead, on the same machine, it is known as a *virtual FTP server* or *virtual host.* This feature lets you run what appear to others as several different FTP servers on one machine. When a remote user uses the virtual FTP server's IP address to access it, the ProFTPD daemon detects that request and operates as the FTP service for that site. ProFTPD can handle a great many virtual FTP sites at the same time on a single machine.

NOTE *Given its configuration capabilities, you can also tailor any of the virtual FTP sites to specific roles, such as a guest site, an anonymous site for a particular group, or an anonymous site for a particular user.*

You configure a virtual FTP server by entering a **<VirtualHost>** directive for it in your **proftpd.conf** file. Such an entry begins with the **VirtualHost** directive and the IP address and ends with a terminating **VirtualHost** directive, **</VirtualHost>**. Any directives placed within these are applied to the virtual host. For anonymous or guest sites, add **Anonymous** and **Guest** directives. You can even add **Directory** directives for specific directories. With the **Port** directive on a standalone configuration, you can create a virtual host that operates on the same system but connects on a different port.

```
<VirtualHost 10.0.0.1>
      ServerName "My virtual FTP server"
</VirtualHost>
```

Configurations for **xinetd** and standalone configurations handle virtual hosts differently. The **xinetd** process detects a request for a virtual host and then hands it off to an FTP daemon. The FTP daemon then examines the address and port specified in the request and processes the request for the appropriate virtual host. In the standalone configuration, the FTP daemon continually listens for requests on all specified ports and generates child processes to handle ones for different virtual hosts as they come in. In the standalone configuration, ProFTPD can support a great many virtual hosts at the same time.

The following example shows a sample configuration of a virtual FTP host. The **VirtualHost** directive uses domain name addresses for its arguments. When a domain name address is used, it must be associated with an IP address in the network's domain name server. The IP address, in turn, has to reference the machine on which the ProFTPD daemon is running. On the **ftp.mypics.com** virtual FTP server, an anonymous guest account named **robpics** is configured that requires a password to log in. An anonymous FTP account is also configured that uses the home directory **/var/ftp/virtual/pics**.

```
<VirtualHost ftp.mypics.com>

  ServerName          "Mypics FTP Server"
  MaxClients          10
  MaxLoginAttempts       1
```

```
  DeferWelcome         on
  <Anonymous ~robpics>
    User         robpics
    Group        robpics
    AnonRequirePassword      on
  <Anonymous /var/ftp/virtual/pics>
    User         ftp
    Group        ftp
    UserAlias    anonymous ftp
  </Anonymous>
</VirtualHost>
```

CHAPTER 23

Web Servers

Linux distributions provide several web servers for use on your system. The primary web server is Apache, which has almost become the standard web server for Linux distributions. It is a very powerful, stable, and fairly easy-to-configure system. Other web servers are also available, such as Tux, which is smaller, but very fast and efficient at handling web data that does not change. Linux distributions provide default configurations for the web servers, making them usable as soon as they are installed.

Apache freely supports full Secure Socket Layer using OpenSSL. There are also private cryptographic products available only with licensing fees. Instead of obtaining the licensing directly, you can simply buy a commercial version of Apache that includes such licensing as Stronghold and Raven (**covalent.net**). Formerly, this kind of restriction applied to the use of RSA technology only in the United States, where it was once patented. The RSA patent has since expired, and RSA is now available for use in freely distributed products like OpenSSL.

Tux

Tux, the Red Hat Content Accelerator, is a static-content web server designed to be run very fast from within the Linux kernel. In effect, it runs in kernel space, making response times much faster than standard user-space web servers like Apache. As a kernel-space server, Tux can handle static content such as images very efficiently. At the same time, it can coordinate with a user-space web server, like Apache, to provide the dynamic content, like CGI programs. Tux can even make use of a cache to hold previously generated dynamic content, using it as if it were static. The ability to coordinate with a user-space web server lets you use Tux as your primary web server. Anything that Tux cannot handle, it will pass off to the user-space web server.

NOTE *Tux is freely distributed under the GNU Public License and is included with many distributions.*

The Tux configuration file is located in **/proc/sys/net/tux**. Here you enter parameters such as `serverport`, `max_doc_size`, and `logfile` (check the Tux reference manual at **redhat.com/docs/manuals/tux** for a detailed listing). Defaults are already entered; `serverport`, `clientport`, and `documentroot` are required parameters that must be set.

serverport is the port Tux will use—80 if it is the primary web server. **clientport** is the port used by the user-space web server Tux coordinates with, such as Apache. **documentroot** specifies the root directory for your web documents (**/var/www/html** on Red Hat and Fedora).

Ideally, Tux is run as the primary web server and Apache as the secondary web server. To configure Apache to run with Tux, the port entry in the Apache **httpd.conf** file needs to be changed from 80 to 8080.

```
Port 8080
```

Several parameters, such as **DOCROOT**, can be specified as arguments to this Tux command. You can enter them in the **/etc/sysconfig/tux** file.

NOTE *You can also run Tux as an FTP server. In the* ***/proc/sys/net/tux*** *directory, you change the contents of the file* ***serverport*** *to 21,* ***application_protocol*** *to 1, and* ***nonagle*** *to 0, and then restart Tux. Use the* ***generatetuxlist*** *command in the document root directory to generate FTP directory listings.*

Alternate Web Servers

Other web servers available for Linux include the Stronghold Enterprise Server and the Apache-SSL server. A listing is provided here:

- Apache-SSL (**apache-ssl.org**) is an encrypting web server based on Apache and OpenSSL (**openssl.org**).
- lighthttpd (**lighttpd.net/**) is a small, very fast, web server.
- Sun Java System web server (**sun.com**) features Java development support and security.
- Zope application server (**zope.org**) is an open source web server with integrated security, web-based administration and development, and database interface features. It was developed by the Zope Corporation, which also developed the Python programming language.
- Stronghold Enterprise Server (**redhat.com/software/stronghold**) is a commercial version of the Apache web server featuring improved security and administration tools.
- Netscape Enterprise Server (**enterprise.netscape.com**), part of Netscape security solutions, features open standards with high performance.
- You can also use the original NCSA web server, though it is no longer under development and is not supported (**hoohoo.ncsa.uiuc.edu**).

Apache Web Server

The Apache web server is a full-featured free HTTP (web) server developed and maintained by the Apache Server Project. The aim of the project is to provide a reliable, efficient, and easily extensible web server, with free open source code made available under its own Apache Software License. The server software includes the server daemon, configuration

files, management tools, and documentation. The Apache Server Project is maintained by a core group of volunteer programmers and supported by a great many contributors worldwide. The Apache Server Project is one of several projects currently supported by the Apache Software Foundation (formerly known as the Apache Group). This nonprofit organization provides financial, legal, and organizational support for various Apache Open Source software projects, including the Apache HTTPD Server, Java Apache, Jakarta, and XML-Apache. The website for the Apache Software Foundation is **apache.org**. Table 23-1 lists various Apache-related websites.

Apache was originally based on the NCSA web server developed at the National Center for Supercomputing Applications, University of Illinois, Urbana-Champaign. Apache has since emerged as a server in its own right and become one of the most popular web servers in use. Although originally developed for Linux and Unix systems, Apache has become a cross-platform application with Windows and OS/2 versions. Apache provides online support and documentation for its web server at **httpd.apache.org**. An HTML-based manual is also provided with the server installation. You can use the Apache Configuration Tool to help configure your Apache server easily. It operates on any X Window System window manager, including GNOME and KDE. In addition, you can use the Comanche configuration tool. Webmin conf also provides Apache configuration support.

Java: Apache Jakarta Project

The Apache Jakarta Project supports the development of Open Source Java software; its website is located at **jakarta.apache.org**. Currently, the Jakarta supports numerous projects, including libraries, tools, frameworks, engines, and server applications. Tomcat is an open source implementation of the Java Servlet and JavaServer Pages specifications. Tomcat is designed for use in Apache servers. JMeter is a Java desktop tool to test performance of server resources, such as servlets and CGI scripts. Velocity is a template engine that provides easy access to Java objects. Watchdog is a tool that checks the compatibility of

Website	Description
apache.org	Apache Software Foundation
httpd.apache.org	Apache HTTP Server Project
jakarta.apache.org	Jakarta Apache Project
apache-gui.com/	Apache GUI Project
comanche.org	Comanche (Configuration Manager for Apache)
apache-ssl.org	Apache-SSL server
openssl.org	OpenSSL project (Secure Socket Layer)
modssl.org	The SSL module (mod_ssl) project to add SSL encryption to an Apache web server
php.net	PHP Hypertext Preprocessor, embedded web page programming language

TABLE 23-1 Apache-Related Websites

servlet containers. Struts, Cactus, and Tapestry are Java frameworks, established methods for developing Java web applications.

Linux Apache Installations

Your Linux distribution will normally provide you with the option of installing the Apache web server during your initial installation of your Linux system. All the necessary directories and configuration files are automatically generated for you. Then, whenever you run Linux, your system is already a fully functional website. Every time you start your system, the web server will also start up, running continuously. On most distributions, the directory reserved for your website data files is **/var/www/html**. Place your web pages in this directory or in any subdirectories. Your system is already configured to operate as a web server. All you need to do is perform any needed network server configurations, and then designate the files and directories open to remote users. You needn't do anything else. Once your website is connected to a network, remote users can access it.

The web server normally sets up your website in the **/var/www** directory. It also sets up several directories for managing the site. The **/var/www/cgi-bin** directory holds the CGI scripts, and **/var/www/html/manual** holds the Apache manual in HTML format. You can use your browser to examine it. Your web pages are to be placed in the **/var/www/html** directory. Place your website home page there. Your configuration files are located in a different directory, **/etc/httpd/conf**. Table 23-2 lists the various Apache web server directories and configuration files.

Website Directories	Description
/var/www/html	Website HTML files
/var/www/cgi-bin	CGI program files
/var/www/html/manual	Apache web server manual
Configuration Files	
.htaccess	Directory-based configuration files; an **.htaccess** file holds directives to control access to files within the directory in which it is located
/etc/httpd/conf	Directory for Apache web server configuration files
/etc/httpd/conf/httpd.conf	Apache web server configuration file
/etc/httpd/conf.d	Directory holding module configuration files such as **ssl.conf** for SSL and **php.conf** for PHP
Application and Module Files	
/usr/sbin	Location of the Apache web server program file and utilities
/usr/share/doc/	Apache web server documentation
/var/log/http	Location of Apache log files
/etc/httpd/modules	Directory holding Apache modules
/etc/httpd/run	Directory holding Apache process IDs

Table 23-2 Apache Web Server Files and Directories (RPM Installation)

Apache Multiprocessing Modules: MPM

Apache now uses a new architecture with multiprocessing modules (MPMs), which are designed to customize Apache to different operating systems, as well as handle certain multiprocessing operations. For the main MPM, a Linux system uses either the prefork or worker MPM, whereas Windows uses the mpm_winnt MPM. The prefork is a standard MPM module designed to be compatible with older Unix and Linux systems, particularly those that do not support threading. You can configure the workload parameters for both in the Apache configuration file, **/etc/httpd/conf/httpd.conf**.

Many directives that once resided in the Apache core are now placed in respective modules and MPMs. With this modular design, several directives have been dropped, such as ServerType. Configuration files for these module are located in the **/etc/httpd/conf.d** directory.

Starting and Stopping the Web Server

On most systems, Apache is installed as a standalone server, continually running. With init scripts, your system automatically starts up the web server daemon, invoking it whenever you start your system.

On Red Hat, Fedora, SUSE, and similar distributions, you can use the **chkconfig** command to set the runlevels at which the httpd server will start, creating links in appropriate runlevel directories. The following command will set up the web server (httpd) to start up at runlevels 3 and 5:

```
chkconfig --level 35 httpd on
```

On Debian, Ubuntu, and similar distributions you can use the use **rrconf** or **sysv-rc-conf** tools. On distributions that use GNOME, you can use the **services-admin** tool.

NOTE *A service script for the web server called* ***httpd (apache2*** *on Debian) is in the* ***/etc/rc.d/init.d*** *directory. You can use the* ***service*** *command to start and stop the httpd server manually:* ***service httpd start****.*

Apache also provides a control tool called **apachectl** (Apache control) for managing your web server. With **apachectl**, you can start, stop, and restart the server from the command line. The **apachectl** command takes several arguments: **start** to start the server, **stop** to stop it, **restart** to shut down and restart the server, and **graceful** to shut down and restart gracefully. In addition, you can use **apachectl** to check the syntax of your configuration files with the **config** argument. You can also use **apachectl** as a system service file for your server in the **/etc/rc.d** directory.

You could call the daemon directly using its full pathname. This daemon has several options. The **-d** option enables you to specify a directory for the **httpd** program if it is different from the default directory. With the **-f** option, you can specify a configuration file different from **httpd.conf**. The **-v** option displays the version.

```
/usr/sbin/httpd -v
```

To check your web server, start your web browser and enter the Internet domain name address of your system. For the system **turtle.mytrek.com**, the user enters **http://turtle.mytrek.com**. This should display the home page you placed in your web root directory.

A simple way to do this is to use Lynx, the command line web browser. Start Lynx and type **g** to open a line where you can enter a URL for your own system. Lynx displays your website's home page. Be sure to place an **index.html** file in the **/var/www/html** directory first.

Once you have your server running, you can check its performance with the **ab** benchmarking tool, also provided by Apache: **ab** shows you how many requests at a time your server can handle. Options include **-v**, which enables you to control the level of detail displayed; **-n**, which specifies the number of requests to handle (default is 1); and **-t**, which specifies a time limit.

NOTE *Currently there is no support for running Apache under* ***xinetd****. In Apache 2, such support is determined by choosing an MPM module designed to run on* ***xinetd****.*

Apache Configuration Files

Configuration directives are placed in the **httpd.conf** configuration file. A documented version of the **httpd.conf** configuration file is installed automatically in **/etc/httpd/conf**. It is strongly recommended that you consult this file on your system. It contains detailed descriptions and default entries for Apache directives.

Any of the directives in the main configuration files can be overridden on a per-directory basis using an **.htaccess** file located within a directory. Although originally designed only for access directives, the **.htaccess** file can also hold any resource directives, enabling you to tailor how web pages are displayed in a particular directory. You can configure access to .**htaccess** files in the **httpd.conf** file.

In addition, many of the modules provided for Apache have their own configuration files. These are placed in the **/etc/httpd/conf.d** directory.

Apache Configuration and Directives

Apache configuration operations take the form of directives entered into the Apache configuration files. With these directives, you can enter basic configuration information, such as your server name, or perform more complex operations, such as implementing virtual hosts. The design is flexible enough to enable you to define configuration features for particular directories and different virtual hosts. Apache has a variety of different directives performing operations as diverse as controlling directory access, assigning file icon formats, and creating log files. Most directives set values such as **DirectoryRoot**, which holds the root directory for the server's web pages, or **Port**, which holds the port on the system that the server listens on for requests. The syntax for a simple directive is shown here:

```
directive option option ...
```

Certain directives create blocks able to hold directives that apply to specific server components (also referred to as sectional directives). For example, the **Directory** directive is used to define a block within which you place directives that apply only to a particular directory. Block directives are entered in pairs: a beginning directive and a terminating directive. The terminating directive defines the end of the block and consists of the same name beginning with a slash. Block directives take an argument that specifies the particular object to which the directives apply. For the **Directory** block directive, you must specify

a directory name to which it will apply. The **<Directory** *mydir***>** block directive creates a block whose directives apply to the *mydir* directory. The block is terminated by a **</Directory>** directive. The **<VirtualHost** *hostaddress***>** block directive is used to configure a specific virtual web server and must include the IP or domain name address used for that server. **</VirtualHost>** is its terminating directive. Any directives you place within this block are applied to that virtual web server. The **<Limit** *method***>** directive specifies the kind of access method you want to limit, such as GET or POST. The access control directives located within the block list the controls you are placing on those methods. The syntax for a block directive is as follows:

```
<block-directive option ... >
 directive option ...
 directive option ...
</block-directive>
```

Usually, directives are placed in one of the main configuration files. Directory directives in those files can be used to configure a particular directory. However, Apache also makes use of directory-based configuration files. Any directory may have its own **.htaccess** file that holds directives to configure only that directory. If your site has many directories, or if any directories have special configuration needs, you can place their configuration directives in their **.htaccess** files, instead of filling the main configuration file with specific **Directory** directives for each one. You can control what directives in an **.htaccess** file take precedence over those in the main configuration files. If your site allows user- or client-controlled directories, you may want to carefully monitor or disable the use of **.htaccess** files in them. (It is possible for directives in an **.htaccess** file to override those in the standard configuration files unless disabled with AllowOverride directives.)

Global Configuration

The standard Apache configuration has three sections: global settings, server settings, and virtual hosts. The global settings control the basic operation and performance of the web server. Here you set configuration locations, process ID files, timing, settings for the MPM module used, and what Apache modules to load.

The **ServerTokens** directive prevents disclosure of any optional modules your server is using. The **ServerRoot** directive specifies where your web server configuration, error, and log files are kept. This is **/etc/httpd**, which will also include your error and log files, as well as the server modules. This server root directory is then used as a prefix to other directory entries.

```
ServerRoot /etc/httpd
```

The server's process ID (PID) file is usually **/etc/httpd/run/httpd.pid**, as set by **PidFile**.

```
PidFile run/httpd.pid
```

Connection and request timing is handled by **Timeout**, **KeepAlive**, **MaxKeepAlive**, and **KeepAliveTimeout** directives. **Timeout** is the time in seconds after which the web server times out a send or receive request. **KeepAlive** allows persistent connections, several requests from a client on the same connection. This is turned off by default.

KeepAliveRequests sets the maximum number of requests on a persistent connection. **KeepAliveTimeout** is the time that a given connection to a client is kept open to receive more requests from that client.

The **Listen** directive will bind the server to a specific port or IP address. By default this is port 80.

```
Listen 80
```

Modules

Much of the power and flexibility of the Apache web server comes from its use of modules to extend its capabilities. Apache is implemented with a core set of directives. Modules can be created that hold definitions of other directives. They can be loaded into Apache, enabling you to use those directives for your server. A standard set of modules is included with the Apache distribution, though you can download others and even create your own. For example, the mod_autoindex module holds the directives for automatically indexing directories (as described in the following section). The mod_mime module holds the MIME type and handler directives. Modules are loaded with the **LoadModule** directive. You can find **LoadModule** directives in the **httpd.conf** configuration file for most of the standard modules.

```
LoadModule mime_module modules/mod_mime.so
```

LoadModule takes as its arguments the name of the module and its location. The modules are stored in the **/etc/httpd/modules** directory, referenced here by the **modules/** prefix.

Configuration files for different modules are located in **/etc/httpd/conf.d** directory. These are also loaded using the **Include** directive. The following inserts all configuration files (those with a **.conf** extension) in the **/etc/httpd/conf.d** directory:

```
Include conf.d/*.conf
```

The apxs application provided with the Apache package can be used to build Apache extension modules. With the apxs application, you can compile Apache module source code in C and create dynamically shared objects that can be loaded with the **LoadModule** directive. The apxs application requires that the mod_so module be part of your Apache application. It includes extensive options such as **-n** to specify the module name, **-a** to add an entry for it in the **httpd.conf** file, and **-i** to install the module on your web server.

You can find a complete listing of Apache web configuration directives at the Apache website, **httpd.apache.org**, and in the Apache manual located in your site's website root directory. On many systems, this is located in the manual subdirectory in the website default directory set up by the distribution (**/var/www/manual**).

MPM Configuration

Configuration settings for MPM prefork and worker modules let you tailor your Apache web server to your workload demands. Default entries will already be set for a standard web server operating under a light load. You can modify these settings for different demands.

Two MPM modules commonly available to Unix and Linux systems are prefork and worker. The prefork module supports one thread per process, which maintains compatibility with older systems and modules. The worker module supports multiple threads for each process, placing a much lower load on system resources. They share several of the same directives, such as **StartServer** and **MaxRequestPerChild**.

Apache runs a single parent process with as many child processes as are needed to handle requests. Configuration for MPM modules focuses on the number of processes that should be available. The prefork module will list server numbers, as a process is started for each server; the worker module will control threads, since it uses threads for each process. The **StartServer** directive lists the number of server processes to start for both modules. This will normally be larger for the prefork than for the worker module.

In the prefork module you need to set minimum and maximum settings for spare servers. **MaxClients** sets the maximum number of servers that can be started, and **ServerLimit** sets the number of servers allowed. The **MaxRequestsPerChild** sets the maximum number of requests allowed for a server.

In the worker module, **MaxClients** also sets the maximum number of client threads, and **ThreadsPerChild** sets the number of threads for each server. **MaxRequestsPerChild** limits the maximum number of requests for a server. Spare thread limits are also configured.

The directives serve as a kind of throttle on the web server access, controlling processes to keep available and limit the resources that can be used. In the prefork configuration, the **StartServer** is set number to 8, and the spare minimum to 5, with the maximum spare as 20. This means that initially 8 server processes will be started up and will wait for requests, along with 5 spare processes. When server processes are no longer being used, they will be terminated until the number of these spare processes is less than 20. The maximum number of server processes that can be started is 256. The maximum number of connections per server process is set at 4,000.

In the worker MPM, only 2 server processes are initially started. Spare threads are set at a minimum of 25 and a maximum of 75. The maximum number of threads is set at 150, with the threads per child at 25.

Server Configuration

Certain directives are used to configure your server's overall operations. These directives are placed midway in the **httpd.conf** configuration file, directly under the section labeled Server Settings. Some directives require pathnames, whereas others only need to be turned on or off with the keywords **on** and **off**. The default **httpd.conf** file already contains these directives. Some are commented out with a preceding **#** symbol. You can activate a directive by removing its **#** sign. Many of the entries are preceded by comments explaining their purpose.

The following is an example of the **ServerAdmin** directive used to set the address where users can send mail for administrative issues. You replace the **you@your.address** entry with the address you want to use to receive system administration mail. By default, this is set to **root@localhost**.

```
# ServerAdmin: Your address, where problems should be e-mailed.
ServerAdmin you@your.address
```

A web server usually uses port 80, which is the Apache default. If you want to use a different port, specify it with the **Port** directive.

```
Port 80
```

The **ServerName** directive holds the hostname for your web server. Specifying a hostname is important to avoid unnecessary DNS lookup failures that can hang your server.

Notice the entry is commented with a preceding **#**. Simply remove the **#** and type your web server's hostname in place of *new.host.name*. If you are using a different port than 80, be sure to specify it attached to the host name, as in **turtle.mytrek.com:80**. Here is the original default entry:

```
# ServerName allows you to set a hostname which is sent
# back to clients for your server if it's different than the
# one the program would get (i.e. use
# "www" instead of the host's real name).

#ServerName new.host.name:80
```

A modified **ServerName** entry would look like this:

```
ServerName turtle.mytrek.com
```

When receiving URL requests for the server system, like those for local files on the system, the **UseCanonicalName** directive will use the **ServerName** and **Port** directives to generate the host URL server name. When off, it will just use the name supplied by the client request. This can be confusing if the web server is referenced by one name but uses another, like **www.mytrek.com** used to reference turtle.mytrek.com. **UseCanonicalName** set to on overcomes this problem, generating the correct local URL.

On Linux systems, entries have already been made for the standard web server installation such as **/var/www** as your website directory. You can tailor your website to your own needs by changing the appropriate directives. The **DocumentRoot** directive determines the home directory for your web pages.

```
DocumentRoot /var/www/html
```

NOTE *You can also configure Apache to operate as just a proxy and/or cache server. Default proxy and cache server directives are already included in the* ***httpd.conf*** *file. The* ***ProxyRequests*** *directive turns proxy activity on. Caching can be configured with directives like* ***CacheRoot*** *to specify the cache directory,* ***CacheSize*** *for the cache size (500KB default), and* ***CacheMaxExpire*** *to set a time limit on unmodified documents.*

Directory-Level Configuration: .htaccess and <Directory>

One of the most flexible aspects of Apache is its ability to configure individual directories. With the **Directory** directive, you can define a block of directives that apply only to a particular directory. Such a directive can be placed in the **httpd.conf** or **access.conf** configuration file. You can also use an **.htaccess** file within a particular directory to hold configuration directives. Those directives are then applied only to that directory. The name ".htaccess" is set with the **AccessFileName** directive. You can change this if you want.

```
AccessFileName .htaccess
```

A Directory block begins with a **<Directory** *pathname***>** directive, where *pathname* is the directory to be configured. The ending directive uses the same **<>** symbols, but with a slash preceding the word "Directory": **</Directory>**. Directives placed within this block apply

only to the specified directory. The following example denies access to the **mypics** directory only to requests from **www.myvids.com**.

```
<Directory /var/www/html/mypics>
 Order Deny,Allow
 Deny from www.myvids.com
</Directory>
```

With the **Options** directive, you can enable certain features in a directory, such as the use of symbolic links, automatic indexing, execution of CGI scripts, and content negotiation. The default is the **All** option, which turns on all features except content negotiation (**Multiviews**). The following example enables automatic indexing (**Indexes**), symbolic links (**FollowSymLinks**), and content negotiation (**Multiviews**).

```
Options Indexes FollowSymLinks Multiviews
```

Configurations made by directives in main configuration files or in upper-level directories are inherited by lower-level directories. Directives for a particular directory held in **.htaccess** files and Directory blocks can be allowed to override those configurations. This capability can be controlled by the **AllowOverride** directive. With the **all** argument, **.htaccess** files can override any previous configurations. The **none** argument disallows overrides, effectively disabling the **.htaccess** file. You can further control the override of specific groups of directives. **AuthConfig** enables use of authorization directives, **FileInfo** is for type directives, **Indexes** is for indexing directives, **Limit** is for access control directives, and **Options** is for the options directive.

```
AllowOverride all
```

Access Control

With access control directives, such as **allow** and **deny**, you can control access to your website by remote users and hosts. The **allow** directive followed by a list of hostnames restricts access to only those hosts. The **deny** directive with a list of hostnames denies access by those systems. The argument **all** applies the directive to all hosts. The **order** directive specifies in what order the access control directives are to be applied. Other access control directives, such as **require**, can establish authentication controls, requiring users to log in. The access control directives can be used globally to control access to the entire site or placed within **Directory** directives to control access to individual directives. In the following example, all users are allowed access:

```
order allow,deny
allow from all
```

URL Pathnames

Certain directives can modify or complete pathname segments of a URL used to access your site. The pathname segment of the URL specifies a particular directory or web page on your site. Directives enable you to alias or redirect pathnames, as well as to select a default web page. With the **Alias** directive, you can let users access resources located in other parts of your system, on other file systems, or on other websites. An alias can use a URL for sites on

the Internet, instead of a pathname for a directory on your system. With the **Redirect** directive, you can redirect a user to another site.

```
Alias /mytrain /home/dylan/trainproj
Redirect /mycars http://www.myautos.com/mycars
```

If Apache is given only a directory to access, rather than a specific web page, it looks for an index web page located in that directory and displays it. The possible names for a default web page are listed by the **DirectoryIndex** directive. The name usually used is **index.html**, but you can add others. The standard names are shown here. When Apache is given only a web directory to access, it looks for and displays the **index.html** web page located in it.

```
DirectoryIndex index.html index.shtml index.cgi
```

Apache also lets a user maintain web pages located in a special subdirectory in the user's home directory, rather than in the main website directory. Using a **~** followed by the username accesses this directory. The name of this directory is specified with the **UserDir** directive. The default name is **public_html**, as shown here. The site **turtle.mytrek.com/~dylan** accesses the directory **turtle.mytrek.com/home/dylan/public_html** on the host **turtle.mytrek.com**.

```
UserDir public_html
```

If you want instead to allow people to use a full pathname, then use a pathname reference. For example, for the user **dylan**, **/usr/www** translates to a URL reference of **/usr/www/dylan**, where the HTML files are located; **/home/*/www** translates to **/home/dylan/www**, a www directory in the user **dylan**'s home directory.

```
UserDir /usr/www
UserDir /home/*/www
```

Userdir access is commented out by default in the standard configuration file. Usually there are users like **root** that you want access denied to. With the disable and enable options, you can open access to certain users while disabling access to others, as shown here:

```
UserDir disable root
UserDir disabled
UserDir enabled dylan chris justin
```

MIME Types

When a browser accesses web pages on a website, it often accesses many different kinds of objects, including HTML files, picture or sound files, and script files. To display these objects correctly, the browser must have some indication of what kind of objects they are. A JPEG picture file is handled differently from a simple text file. The server provides this type information in the form of MIME types. MIME types are the same types used for sending attached files through Internet mailers, such as Pine. Each kind of object is associated with a given MIME type. Provided with the MIME type, the browser can correctly handle and display the object.

The MIME protocol associates a certain type with files of a given extension. For example, files with a **.jpg** extension have the MIME type image/jpeg. The **TypesConfig** directive holds the location of the **mime.types** file, which lists all the MIME types and their associated file extensions. **DefaultType** is the default MIME type for any file whose type cannot be determined. **AddType** enables you to modify the **mime.type** types list without editing the MIME file.

```
TypesConfig /etc/mime.types
DefaultType text/plain
```

Other type directives are used to specify actions to be taken on certain documents. **AddEncoding** lets browsers decompress compressed files on the fly. **AddHandler** maps file extensions to actions, and **AddLanguage** enables you to specify the language for a document. The following example marks filenames with the **.gz** extension as gzip-encoded files and files with the **.fr** extension as French language files:

```
AddEncoding x-gzip gz
AddLanguage fr .fr
```

A web server can display and execute many different types of files and programs. Not all web browsers are able to display all those files, though. Older browsers are the most limited. Some browsers, such as Lynx, are not designed to display even simple graphics. To allow a web browser to display a page, the server negotiates with it to determine the type of files it can handle. To enable such negotiation, you need to enable the **Multiviews** option.

```
Option multiviews
```

CGI Files

Common Gateway Interface (CGI) files are programs that can be executed by web browsers accessing your site. CGI files are usually initiated by web pages that execute the program as part of the content they display. Traditionally, CGI programs were placed in a directory called **cgi-bin** and could be executed only if they resided in such a special directory. Usually, only one **cgi-bin** directory exists per website. Distributions will normally set up a **cgi-bin** directory in the default web server directory (**/var/www/cgi-bin** on Fedora). Here, you place any CGI programs that can be executed on your website. The **ScriptAlias** directive specifies an alias for your **cgi-bin** directory. Any web pages or browsers can use the alias to reference this directory.

```
ScriptAlias /cgi-bin/ /var/www/cgi-bin/
```

Automatic Directory Indexing

When given a URL for a directory instead of an HTML file, and when no default web page is in the directory, Apache creates a page on the fly and displays it. This is usually only a listing of the different files in the directory. In effect, Apache indexes the items in the directory for you. You can set several options for generating and displaying such an index. If **FancyIndexing** is turned on, web page items are displayed with icons and column headers that can be used to sort the listing.

```
FancyIndexing on
```

Authentication

Your web server can also control access on a per-user or per-group basis to particular directories on your website. You can require various levels for authentication. Access can be limited to particular users and require passwords, or expanded to allow members of a group access. You can dispense with passwords altogether or set up an anonymous type of access, as used with FTP.

To apply authentication directives to a certain directory, you place those directives within either a **Directory** block or the directory's **.htaccess** file. You use the **require** directive to determine what users can access the directory. You can list particular users or groups. The **AuthName** directive provides the authentication realm to the user, the name used to identify the particular set of resources accessed by this authentication process. The **AuthType** directive specifies the type of authentication, such as basic or digest. A **require** directive requires also **AuthType**, **AuthName**, and directives specifying the locations of group and user authentication files. In the following example, only the users **george**, **robert**, and **mark** are allowed access to the **newpics** directory:

```
<Directory /var/www/html/newpics
     AuthType Basic
     AuthName Newpics
     AuthUserFile /web/users
     AuthGroupFile /web/groups
     <Limit GET POST>
           require users george robert mark
     </Limit>
</Directory>
```

The next example allows group access by administrators to the CGI directory:

```
<Directory /var/www/html/cgi-bin
     AuthType Basic
     AuthName CGI
     AuthGroupFile /web/groups
     <Limit GET POST>
           require groups admin
     </Limit>
s</Directory>
```

To set up anonymous access for a directory, place the **Anonymous** directive with the user anonymous as its argument in the directory's Directory block or **.htaccess** file. You can also use the **Anonymous** directive to provide access to particular users without requiring passwords from them.

Apache maintains its own user and group authentication files specifying what users and groups are allowed access to which directories. These files are normally simple flat files, such as your system's password and group files. They can become large, however, possibly slowing down authentication lookups. As an alternative, many sites have used database management files in place of these flat files. Database methods are then used to access the files, providing a faster response time. Apache has directives for specifying the authentication files, depending on the type of file you are using. The **AuthUserfile** and **AuthGroupFile** directives are used to specify the location of authentication files that have

a standard flat-file format. The **AuthDBUserFile** and **AuthDBGroupFile** directives are used for DB database files, and the **AuthDBMGUserFIle** and **AuthDBMGGroupFile** are used for DBMG database files.

The programs htdigest, htpasswd, and dbmmanage are tools provided with the Apache software package for creating and maintaining *user authentication files,* which are user password files listing users who have access to specific directories or resources on your website. The htdigest and htpasswd programs manage a simple flat file of user authentication records, whereas dbmmanage uses a more complex database management format. If your user list is extensive, you may want to use a database file for fast lookups. htdigest takes as its arguments the authentication file, the realm, and the username, creating or updating the user entry. htpasswd can also employ encryption on the password. dbmmanage has an extensive set of options to add, delete, and update user entries. A variety of different database formats are used to set up such files. Three common ones are Berkeley DB2, NDBM, and GNU GBDM. dbmmanage looks for the system libraries for these formats in that order. Be careful to be consistent in using the same format for your authentication files.

Log Files

Apache maintains logs of all requests by users to your website. By default, these logs include records using the Common Log Format (CLF). The record for each request takes up a line composed of several fields: host, identity check, authenticated user (for logins), the date, the request line submitted by the client, the status sent to the client, and the size of the object sent in bytes.

Webalizer

Reports on web logs can be generated using the Webalizer tool. Webalizer will display information on your website usage. When you run the **webalizer** command, usage reports will be placed in the **/var/www/html/usage** directory. Access the index page to display a page with links to monthly reports, **file:/var/www/html/usage/index.html**. Report configuration is specified in the **/etc/webalizer.conf** file. Previous summaries are kept in the **/etc/webalizer .history** file.

Customizing Logs

Using the **LogFormat** and **CustomLog** directives, you can customize your log record to add more fields with varying levels of detail. These directives use a format string consisting of field specifiers to determine the fields to record in a log record. You add whatever fields you want and in any order. A field specifier consists of a percent (**%**) symbol followed by an identifying character. For example, **%h** is the field specifier for a remote host, **%b** for the size in bytes, and **%s** for the status. See the documentation for the mod_log_config module for a complete listing. You should quote fields whose contents may take up more than one word. The quotes themselves must be quoted with a backslash to be included in the format string. The following example is the Common Log Format implemented as a **FormatLog** directive:

```
FormatLog "%h %l %u %t \"%r\" %s %b"
```

Certain field specifiers in the log format can be qualified to record specific information. The **%i** specifier records header lines in requests the server receives. The reference for the specific header line to record is placed within braces between the **%** and the field specifier.

For example, **User-agent** is the header line that indicates the browser software used in the request. To record User-agent header information, use the conversion specifier **%{User-agent}i**.

To maintain compatibility with NCSA servers, Apache originally implemented **AgentLog** and **RefererLog** directives to record User-agent and Referer headers. These have since been replaced by qualified **%i** field specifiers used for the **LogFormat** and **CustomLog** directives. A Referer header records link information from clients, detecting who may have links to your site. The following is an NCSA-compliant log format:

```
"%h %l %u %t \"%r\" %s %b\"%{Referer}i\" \"%{User-agent}i\"".
```

Generating and Managing Log Files

Instead of maintaining one large log file, you can create several log files using the **CustomLog** or **TransferLog** directive. This is helpful for virtual hosts where you may want to maintain a separate log file for each host. You use the **FormatLog** directive to define a default format for log records. The **TransferLog** then uses this default as its format when creating a new log file. **CustomLog** combines both operations, enabling you to create a new file and to define a format for it.

```
FormatLog "%h %l %u %t \"%r\" %s %b"
# Create a new log file called myprojlog using the FormatLog format
TransferLog myprojlog
# Create a new log file called mypicslog using its own format
CustomLog mypicslog "%h %l %u %t \"%r\" %s %b"
```

Apache provides two utilities for processing and managing log files: **logresolve** resolves IP addresses in your log file to hostnames; **rotatelogs** rotates log files without having to kill the server. You can specify the rotation time.

NOTE *The Apache web server can also provide detailed reports on server activity and configuration, letting you display this information to remote servers. The* ***Location*** *directive server-info will display the configuration details of your web server, and the server-status directive will show web processes. The pages server-info and server-status will display the reports, as in* ***http://localhost/server-info****. Use the ExtendedStatus directive to enable detailed reports.*

Virtual Hosting on Apache

Virtual hosting allows the Apache web server to host multiple websites as part of its own. In effect, the server can act as several servers, each hosted website appearing separate to outside users. Apache supports both IP address–based and name-based virtual hosting. IP address–based virtual hosts use valid registered IP addresses, whereas name-based virtual hosts use fully qualified domain addresses. These domain addresses are provided by the host header from the requesting browser. The server can then determine the correct virtual host to use on the basis of the domain name alone. Note that SSL servers require IP virtual hosting. See **httpd.apache.org** for more information.

IP-Based Virtual Hosting

In the IP address–based virtual hosting method, your server must have a different IP address for each virtual host. The IP address you use is already set up to reference your system. Network system administration operations can set up your machine to support several IP addresses. Your machine can have separate physical network connections for each one, or a particular connection can be configured to listen for several IP addresses at once. In effect, any of the IP addresses can access your system.

You can configure Apache to run a separate daemon for each virtual host, separately listening for each IP address, or you can have a single daemon running that listens for requests for all the virtual hosts. To set up a single daemon to manage all virtual hosts, use **VirtualHost** directives. To set up a separate daemon for each host, also use the **Listen** directive.

Name-Based Virtual Hosting

With IP-based virtual hosting, you are limited to the number of IP addresses your system supports. With name-based virtual hosting, you can support any number of virtual hosts using no additional IP addresses. With only a single IP address for your machine, you can still support an unlimited number of virtual hosts. Such a capability is made possible by the HTTP/1.1 protocol, which lets a server identify the name by which it is being accessed. This method requires the client, the remote user, to use a browser that supports the HTTP/1.1 protocol, as current browsers do (though older ones may not). A browser using such a protocol can send a host header specifying the particular host to use on a machine.

If your system has only one IP address, implementing virtual hosts prevents access to your main server with that address. You can no longer use your main server as a web server directly; you can use it only indirectly to manage your virtual host. However, you can configure a virtual host to manage your main server's web pages. You then use your main server to support a set of virtual hosts that would function as websites, rather than the main server operating as one site directly. If your machine has two or more IP addresses, you can use one for the main server and the other for your virtual hosts. You can even mix IP-based virtual hosts and name-based virtual hosts on your server. You can also use separate IP addresses to support different sets of virtual hosts. You can further have several domain addresses access the same virtual host. To do so, place a **ServerAlias** directive listing the domain names within the selected **VirtualHost** block.

```
ServerAlias www.mypics.com www.greatpics.com
```

Requests sent to the IP address used for your virtual hosts have to match one of the configured virtual domain names. To catch requests that do not match one of these virtual hosts, you can set up a default virtual host using *_default_*:*. Unmatched requests are then handled by this virtual host.

```
<VirtualHost _default_:*>
```

Dynamic Virtual Hosting

If you have implemented many virtual hosts on your server that have the same configuration, you can use a technique called *dynamic virtual hosting* to have these virtual hosts generated dynamically. The code for implementing your virtual hosts becomes much smaller, and as

a result, your server accesses them faster. Adding yet more virtual hosts becomes a simple matter of creating appropriate directories and adding entries for them in the DNS server.

To make dynamic virtual hosting work, the server uses commands in the mod_vhost_alias module (supported in Apache version 1.3.6 and up) to rewrite both the server name and the document root to those of the appropriate virtual server (for older Apache versions before 1.3.6, you use the mod_rewrite module). Dynamic virtual hosting can be either name-based or IP-based. In either case, you have to set the **UseCanonicalName** directive in such a way as to allow the server to use the virtual hostname instead of the server's own name. For name-based hosting, you simply turn off **UseCanonicalName**. This allows your server to obtain the hostname from the host header of the user request. For IP-based hosting, you set the **UseCanonicalName** directive to DNS. This allows the server to look up the host in the DNS server.

```
UseCanonicalName Off
UseCanonicalName DNS
```

You then have to enable the server to locate the different document root directories and CGI bin directories for your various virtual hosts. You use the **VirtualDocumentRoot** directive to specify the template for virtual hosts' directories. For example, if you place the different host directories in the **/var/www/hosts** directory, you can then set the **VirtualDocumentRoot** directive accordingly.

```
VirtualDocumentRoot /var/www/hosts/%0/html
```

The %0 will be replaced with the virtual host's name when that virtual host is accessed. It is important that you create the dynamic virtual host's directory using that host's name. For example, for a dynamic virtual host called **www.mygolf.org**, you first create a directory named **/var/www/hosts/www.mygolf.org**, then create subdirectories for the document root and CGI programs, as in **/var/www/hosts/www.mygolf.org/html**. For the CGI directory, use the **VirtualScriptAlias** directive to specify the CGI subdirectory you use.

```
VirtualScriptAlias /var/www/hosts/%0/cgi-bin
```

A simple example of name-based dynamic virtual hosting directives follows:

```
UseCanonicalName Off
VirtualDocumentRoot /var/www/hosts/%0/html
VirtualScriptAlias /var/www/hosts/%0/cgi-bin
```

A request for **www.mygolf.com/html/mypage** evaluates to

```
/var/www/hosts/www.mygolf.com/html/mypage
```

A simple example of dynamic virtual hosting is shown here:

```
UseCanonicalName Off

NameVirtualHost 192.168.1.5

<VirtualHost 192.168.1.5>
 ServerName www.mygolf.com
```

```
 ServerAdmin webmaster@mail.mygolf.com
 VirtualDocumentRoot /var/www/hosts/%0/html
 VirtualScriptAlias /var/www/hosts/%0/cgi-bin
 ...
</VirtualHost>
```

To implement IP-based dynamic virtual hosting instead, set the **UseCanonicalName** to DNS instead of Off.

```
UseCanonicalName DNS
VirtualDocumentRoot /var/www/hosts/%0/html
VirtualScriptAlias /var/www/hosts/%0/cgi-bin
```

Interpolated Strings

The mod_vhots_alias module supports various interpolated strings, each beginning with a **%** symbol and followed by a number. As you have seen, **%0** references the entire web address. **%1** references only the first segment, **%2** references the second, **%-1** references the last part, and **%2+** references from the second part on. For example, to use only the second part of a web address for the directory name, use the following directives:

```
VirtualDocumentRoot /var/www/hosts/%2/html
VirtualScriptAlias /var/www/hosts/%2/cgi-bin
```

In this case, a request made for **www.mygolf.com/html/mypage** uses only the second part of the web address. This would be "mygolf" in **www.mygolf.com**, and would evaluate to

```
/var/www/hosts/mygolf/html/mypage
```

If you used **%2+** instead, as in **/var/www/hosts/%2/html**, the request for **www.mygolf.com/html/mypage** would evaluate to

```
/var/www/hosts/mygolf.com/html/mypage
```

The same method works for IP addresses, where **%1** references the first IP address segment, **%2** references the second, and so on.

Logs for Virtual Hosts

One drawback of dynamic virtual hosting is that you can set up only one log for all your hosts. However, you can create your own shell program to simply cut out the entries for the different hosts in that log.

```
LogFormat "%V %h %l %u %t \"%r\" %s %b" vcommon
CustomLog logs/access_log vcommon
```

IP Addressing

Implementing dynamic virtual hosting in the standard way as shown previously will slow down the process, as your server will have to perform a DNS lookup to discover the name of your server using its IP address. You can avoid this step by simply using the IP address for your virtual host's directory. So, for IP virtual host 192.198.1.6, you create a directory

/var/www/hosts/192.198.1.6, with an **html** subdirectory for that host's document root. You use the **VirtualDocumentRootIP** and **VirtualScriptAliasIP** directives to use IP addresses as directory names. Now the IP address can be mapped directly to the document root directory name, no longer requiring a DNS lookup. Also, be sure to include the IP address in your log, **%A**.

```
UseCanonicalName DNS
LogFormat "%A %h %l %u %t \"%r\" %s %b" vcommon
CustomLog logs/access_log vcommon
VirtualDocumentRootIP /var/www/hosts/%0/html
VirtualScriptAliasIP /var/www/hosts/%0/cgi-bin
```

Server-Side Includes

Server-side includes (SSIs) are designed to provide a much more refined control of your website content, namely the web pages themselves. Server-side includes are Apache directives placed within particular web pages as part of the page's HTML code. You can configure your Apache web server to look for SSI directives in particular web pages and execute them. First, you have to use the **Options** directive with the **include** option to allow SSI directives.

```
Options Includes
```

You need to instruct the server to parse particular web pages. The easiest way to enable parsing is to instruct Apache to parse HTML files with specified extensions. Usually, the extension **.shtml** is used for web pages that have SSI directories. In fact, in the default Apache configuration files, you can find the following entry to enable parsing for SSI directives in HTML files. The **AddType** directive here adds the **.shtml** type as an HTML type of file, and the **AddHandler** directive specifies that **.shtml** files are to be parsed (server-parsed):

```
# To use server-parsed HTML files
AddType text/html .shtml
AddHandler server-parsed .shtml
```

Instead of creating a separate type of file, you can use the **XBitHack** directive to have Apache parse any executable file for SSI directives. In other words, any file with execute permission will be parsed for SSI directives.

SSI directives operate much like statements in a programming language. You can define variables, create loops, and use tests to select alternate directives. An SSI directive consists of an element followed by attributes that can be assigned values. The syntax for an SSI directive is shown here:

```
<!--#element attribute=value ... -->
```

You can think of an element as operating much like a command in a programming language and attributes as its arguments. For example, to assign a value to a variable, you use the **set** element with the variable assignment as its attribute. The **if** directive displays

any following text on the given web page. The **if** directive takes as its attribute **expr**, which is assigned the expression to test. The test is able to compare two strings using standard comparison operators such as **<=**, **!=**, or **=**. Variables used in the test are evaluated with the **$** operator.

```
<!--#set myvar="Goodbye" -->
<!--#if expr="$myvar = Hello" -->
```

Other helpful SSI elements are **exec**, which executes CGI programs, or shell commands, which read the contents of a file into the web page and also execute CGI files. The **echo** element displays values such as the date, the document's name, and the page's URL. With the **config** element, you can configure certain values, such as the date or file size.

PHP

PHP (PHP: Hypertext Preprocessor) is a scripting language designed for use in web pages. PHP-enabled pages allow you to create dynamic web pages that can perform tasks instead of just displaying data. PHP is an official project of the Apache Software Foundation. You can find out more about PHP at **php.net**.

Unlike CGI programs, which are executed separately from a web page, PHP commands are embedded as tags within the page itself, much as SSI commands are. PHP support to interpret and execute these commands is provided directly by the web server. This embedded support is enabled in Apache with the mod_php module (**/etc/httpd/conf.d/php.conf** configuration file). That is, instead of having to separately construct programs to be invoked and run outside the web server, in PHP such commands are embedded within a web page and run by the web server. The web server maintains complete control at all times whenever tasks are being performed. It is possible, however, to implement PHP in a CGI mode, where PHP pages are constructed as separate programs invoked by a web page, much as a Perl-based CGP program is.

PHP has flexible and powerful programming capabilities on the same level as C and Perl. As in those languages, you can create control structures such as if statements and loops. In addition, PHP has capabilities specifically suited to web page tasks. PHP can interact directly with databases such as Oracle, MySQL, and IBM DB2. It can easily interact with all the standard protocols, such as IMAP, LDAP, HTTP, and POP3. It even has text processing abilities such as interpreting regular expressions and displaying XML documents. There are also extensions for searches, compression tools like gzip, and language translations. PHP supports a massive collection of possible operations. Check its website for a complete listing, as well as online manuals and tutorials.

Apache Configuration Tool

The Apache Configuration Tool opens with a window displaying panels for Main, Virtual Hosts, Server, and Performance Tuning. In each of these you will see buttons to open dialog boxes where you can enter default settings. You will also be able to enter settings for particular items such as virtual hosts and directories. For example, in the Virtual Hosts panel you can enter default settings for all virtual hosts, as well as add and edit particular

virtual hosts. Click the Help button to display a web page–based reference manual that details how to use each panel.

- On the Main panel, you enter your web server address, the web master's email address, and the ports the web server will be listening on.
- On the Virtual Hosts panel, be sure to select Default Virtual Host and click Edit to set the default settings for server options, pages searches, SSL support, log files, CGI environment support, and directories (Performance). To add a virtual host, click Add to open a window where you can enter host information such as the virtual hostname and IP address. You can select different configuration panels for the virtual host, such as log files and directory controls.
- On the server panel, you set administrative settings such as the Apache server's user ID and the process ID file, along with the user and group.
- The Performance Tuning panel lets you set different usage limits such as the maximum number of requests and the number of requests per connection.

When the Apache Configuration Tool saves its settings, it will overwrite the Apache configuration file, **/etc/httpd/conf/httpd.conf**. It is advisable that you first make a backup copy of your **httpd.conf** file in case you want to restore the original settings created by your distribution for Apache. If you have already manually edited this file, you will receive a warning, and the Apache Configuration Tool will make a backup copy in **/etc/httpd/conf/httpd.conf.bak**.

Web Server Security: SSL

Web server security deals with two different tasks: protecting your web server from unauthorized access, and providing security for transactions carried out between a web browser client and your web server. To protect your server from unauthorized access, you use a proxy server such as Squid. Squid is a GNU proxy server often used with Apache on Linux systems. Apache itself has several modules that provide security capabilities. These include mod_access for mandatory controls; mod_auth, mod_auth_db, mod_auth_digest, and mod_auth_dbm, which provide authentication support; and mod_auth_anon for anonymous FTP-like logging (see previous sections on access control and authentication).

To secure transmissions, you need to perform three tasks. You have to verify identities, check the integrity of the data, and ensure the privacy of the transmission. To verify the identities of the hosts participating in the transmission, you perform authentication procedures. To check the integrity of the data, you add digital signatures containing a digest value for the data. The digest value is a value that uniquely represents the data. Finally, to secure the privacy of the transmission, you encrypt it. Transactions between a browser and your server can then be encrypted, with the browser and your server alone able to decrypt the transmissions. The protocol most often used to implement secure transmissions with Linux Apache web servers is the Secure Sockets Layer (SSL) protocol, which was originally developed by Netscape for secure transactions on the web.

Like the Secure Shell (SSH) and the GNU Privacy Guard, SSL uses a form of public- and private-key encryption for authentication. Data is encrypted with the public key but can be decrypted only with the private key. Once the data is authenticated, an agreed-upon cipher

is used to encrypt it. Digital signatures encrypt an MD5 digest value for data to ensure integrity. Authentication is carried out with the use of certificates of authority. Certificates identify the different parties in a secure transmission, verifying that they are who they say they are. A web server will have a certificate verifying its identity, verifying that it is the server it claims to be. The browser contacting the server will also have a certificate identifying who it is. These certificates are, in turn, both signed by a certificate authority, verifying that they are valid certificates. A certificate authority is an independent entity that both parties trust.

A certificate contains the public key of the particular server or browser it is given to, along with the digital signature of the certificate authority and identity information such as the name of the user or company running the server or browser. The effectiveness of a certificate depends directly on the reliability of the certificate authority issuing it. To run a secure web server on the Internet, you should obtain a certificate from a noted certificate authority such as VeriSign. A commercial vendor such as Stronghold can do this for you. Many established companies already maintain their own certificate authority, securing transmissions within their company networks. An SSL session is set up using a handshake sequence in which the server and browser are authenticated by exchanging certificates, a cipher is agreed upon to encrypt the transmissions, and the kind of digest integrity check is chosen. There is also a choice in the kind of public-key encryption used for authentication, either RSA or DSA. For each session, a unique session key is set up that the browser and server use.

A free open source version of SSL called OpenSSL is available for use with Apache (see **openssl.org**). It is based on SSLeay from Eric A. Young and Tim J. Hudson. However, U.S. government restrictions prevent the Apache web server from being freely distributed with SSL capabilities built in. You have to separately obtain SSL and update your Apache server to incorporate this capability.

The U.S. government maintains export restrictions on encryption technology over 40 bits. SSL, however, supports a number of ciphers using 168-, 128-, and 40-bit keys (128 is considered secure, so by comparison the exportable 40-bit versions are useless). This means that if Apache included SSL, it could not be distributed outside the United States. Outside the United States, however, there are projects that do distribute SSL for Apache using OpenSSL. These are free for noncommercial use in the United States, though export restrictions apply. The Apache-SSL project freely distributes Apache with SSL built in, apache+ssl. You can download this from their website at **apache-ssl.org** (though there are restrictions on exporting encryption technology, there are none on importing it). In addition, the mod_ssl project provides an SSL module with patches you can use to update your Apache web server to incorporate SSL (**modsll.org**). mod_ssl is free for both commercial and noncommercial use under an Apache-style license (**/etc/httpd/conf.d/ssl.conf** configuration file).

The mod_ssl implementation of SSL provides an alternate access to your web server using a different port (443) and a different protocol, https. In effect, you have both an SSL server and a nonsecure version. To access the secure SSL version, you use the protocol https instead of http for the web server's URL address. For example, to access the SSL version for the web server running at **mytrek.com**, you use the protocol https in its URL, as shown here:

```
https://www.mytrek.com
```

You can configure mod_ssl using a number of configuration directives in the Apache configuration file, **smb.conf**. The default configuration file installed with Apache contains a section for the SSL directives along with detailed comments. Check the online documentation for mod_ssl at **modssl.org** for a detailed reference listing all the directives. There are global, server-based, and directory-based directives available.

In the **smb.conf** file, the inclusion of SSL directives are controlled by IfDefine blocks enabled by the HAVE_SSL flag. For example, the following code will load the SSL module:

```
<IfDefine HAVE_SSL>
LoadModule ssl_module     modules/libssl.so
</IfDefine>
```

The SSL version for your Apache web server is set up in the **smb.conf** file as a virtual host. The SSL directives are enabled by an ifDefine block using the HAVE_SSL flag. Several default directives are implemented, such as the location of SSL key directories and the port that the SSL version of the server will listen on (443). Others are commented out. You can enable them by removing the preceding # symbol, setting your own options. Several of the directives are shown here:

```
<IfDefine HAVE_SSL>
## SSL Virtual Host Context

#  Server Certificate:
SSLCertificateFile /etc/httpd/conf/ssl.crt/server.crt

#  Server Private Key:
SSLCertificateKeyFile /etc/httpd/conf/ssl.key/server.key

#  Certificate Authority (CA):
#SSLCACertificatePath /etc/httpd/conf/ssl.crt
#SSLCACertificateFile /etc/httpd/conf/ssl.crt/ca-bundle.crt
```

In the **/etc/httpd/conf** directory, mod_ssl will set up several SSL directories that contain SSL authentication and encryption keys and data. The **ssl.crt** directory will hold certificates for the server. The **ssl.key** directory holds the public and private keys used in authentication encryption. Revocation lists for revoking expired certificates are kept in **ssl.crl**. The **ssl.csr** directory holds the certificate signing request used to request an official certificate from a certificate authority. **ssl.prm** holds parameter files used by the DSA key encryption method. Check the **readme** files in each directory for details on the SSL files they contain.

The mod_ssl installation will provide you with a demonstration certificate called snakeoil that you can use to test your SSL configuration. When you have an official certificate, you can install it with the **`make certificate`** command within the **ssl.crt** directory. This will overwrite the **server.crt** server certificate file.

24 CHAPTER

Proxy Servers

Proxy servers operate as an intermediary between a local network and services available on a larger one such as the Internet. Requests from local clients for web services can be handled by the proxy server, speeding transactions as well as controlling access. Proxy servers maintain current copies of commonly accessed web pages, speeding web access times by eliminating the need to access the original site constantly. They also perform security functions, protecting servers from unauthorized access. *Squid* is a free, open source, proxy-caching server for web clients, designed to speed Internet access and provide security controls for web servers. It implements a proxy-caching service for web clients that caches web pages as users make requests. Copies of web pages accessed by users are kept in the Squid cache, and as requests are made, Squid checks to see if it has a current copy. If Squid does have a current copy, it returns the copy from its cache instead of querying the original site. If it does not have a current copy, it will retrieve one from the original site. Replacement algorithms periodically replace old objects in the cache. In this way, web browsers can then use the local Squid cache as a proxy HTTP server. Squid currently handles web pages supporting the HTTP, FTP, and SSL protocols (Squid cannot be used with FTP clients), each with an associated default port (see Table 24-1). It also supports ICP (Internet Cache Protocol), HTCP (Hypertext Caching Protocol) for web caching, and SNMP (Simple Network Management Protocol) for providing status information.

You can find out more about Squid at **squid-cache.org**. For detailed information, check the Squid FAQ and the user manual located at their website. The FAQ is also installed in your **/usr/share/doc** under the **squid** directory.

As a proxy, Squid does more that just cache web objects. It operates as an intermediary between the web browsers (clients) and the servers they access. Instead of connections being made directly to the server, a client connects to the proxy server. The proxy then relays requests to the web server. This is useful for situations where a web server is placed behind a firewall server, protecting it from outside access. The proxy is accessible on the firewall, which can then transfer requests and responses back and forth between the client and the web server. The design is often used to allow web servers to operate on protected local networks and still be accessible on the Internet. You can also use a Squid proxy to provide web access to the Internet by localhosts. Instead of using a gateway providing complete access to the Internet, localhosts can use a proxy to allow them just web access (see Chapter 5). You can also combine the two, allowing gateway access, but using the proxy server to provide more control for web access. In addition, the caching capabilities of Squid can provide localhosts with faster web access.

Protocol	Description and Port
HTTP	Web pages, port 3128
FTP	FTP transfers through websites, port 3128
ICP	Internet Caching Protocol, port 3130
HTCP	Hypertext Caching Protocol, port 4827
CARP	Cache Array Routing Protocol
SNMP	Simple Network Management Protocol, port 3401
SSL	Secure Socket Layer

Table 24-1 Protocols Supported by Squid

Technically, you could use a proxy server to simply manage traffic between a web server and the clients that want to communicate with it, without doing caching at all. Squid combines both capabilities as a proxy-caching server.

Squid also provides security capabilities that let you exercise control over hosts accessing your web server. You can deny access by certain hosts and allow access by others. Squid also supports the use of encrypted protocols such as SSL (see Chapter 23). Encrypted communications are tunneled (passed through without reading) through the Squid server directly to the web server.

Squid is supported and distributed under a GNU Public License by the National Laboratory for Applied Network Research (NLANR) at the University of California, San Diego. The work is based on the Harvest Project to create a web indexing system that includes a high-performance cache daemon called **cached**. You can obtain current source code versions and online documentation from the Squid home page at **squid-cache.org**. The Squid software package consists of the Squid server, several support scripts for services like LDAP and HTTP, and a cache manager script called `cachemgr.cgi`. The `cachemgr.cgi` lets you view statistics for the Squid server as it runs. You can set the Squid server to start up automatically using `chkconfig`, GNOME's **services-admin**, Debian's **rcconf**, or **sysv-rc-conf**.

Configuring Client Browsers

Squid supports both standard proxy caches and transparent caches. With a standard proxy cache, users will need to configure their browsers to specifically access the Squid server. A transparent cache, on the other hand, requires no browser configuration by users. The cache is transparent, allowing access as if it were a normal website. Transparent caches are implemented by IPtables using net filtering to intercept requests and direct them to the proxy cache (see Chapter 20).

With a standard proxy cache, users need to specify their proxy server in their web browser configuration. For this they will need the IP address of the host running the Squid proxy server as well as the port it is using. Proxies usually make use of port 3128. To configure use of a proxy server running on the local sample network described in Chapter 5, you enter the following. (the proxy server is running on **turtle.mytrek.com** (192.168.0.1) and using port 3128):

```
192.168.0.1 3128
```

On Firefox, Mozilla, and Netscape, the user on the sample local network first selects the Proxy panel located in Preferences under the Edit menu. Then, in the Manual proxy configuration's View panel, you enter the previous information. The user will see entries for FTP, HTTP, and security proxies. For standard web access, enter the IP address in the FTP and web boxes. For their port boxes, enter **3128**.

For GNOME, select Network Proxy in the Preferences menu or window, and for Konqueror on the KDE Desktop, select the Proxies panel on the Preferences | Web Browsing menu window. Here, you can enter the proxy server address and port numbers. If your localhost is using Internet Explorer (as a Windows system does), you set the proxy entries in the Local Area Network settings accessible from the Internet Options window.

On Linux or Unix systems, localhosts can set the **http_proxy** and **ftp_proxy** shell variables to configure access by Linux-supported web browsers such as Lynx. You can place these definitions in your **.bash_profile** or **/etc/profile** file to have them automatically defined whenever you log in.

```
http_proxy=192.168.0.1:3128
ftp proxy=192.168.0.1:3128
export http_proxy ftp_proxy
```

Alternatively, you can use the proxy's URL.

```
http_proxy=http://turtle.mytrek.com:3128
```

For the Elinks browser, you can specify a proxy in its configuration file, **/etc/elinks.conf**. Set both FTP and web proxy host options, as in:

```
protocol.http.proxy.host  turtle.mytrek.com:3128
protocol.ftp.proxy.host  turtle.mytrek.com:3128
```

Before a client on a localhost can use the proxy server, access permission has to be given to it in the server's **squid.conf** file, described in the later section "Security." Access can easily be provided to an entire network. For the sample network used here, you would have to place the following entries in the **squid.conf** file. These are explained in detail in the following sections.

```
acl mylan src 192.168.0.0/255.255.255.0
http_access allow mylan
```

TIP *Web clients that need to access your Squid server as a standard proxy cache will need to know the server's address and the port for Squid's HTTP services, by default 3128.*

The squid.conf File

The Squid configuration file is **squid.conf**, located in the **/etc/squid** directory. In the **/etc/squid/squid.conf** file, you set general options such as ports used, security options controlling access to the server, and cache options for configuring caching operations. You can use a backup version called **/etc/squid/squid.conf.default** to restore your original defaults. The default version of **squid.conf** provided with Squid software includes detailed

explanations of all standard entries, along with commented default entries. Entries consist of tags that specify different attributes. For example, **maximum_object_size** and **maximum_object** set limits on objects transferred.

```
maximum_object_size 4096 KB
```

As a proxy, Squid will use certain ports for specific services, such as port 3128 for HTTP services like web browsers. Default port numbers are already set for Squid. Should you need to use other ports, you can set them in the **/etc/squid/squid.conf** file. The following entry shows how you set the web browser port:

```
http_port 3128
```

NOTE *Squid uses the Simple Network Management Protocol (SNMP) to provide status information and statistics to SNMP agents managing your network. You can control SNMP with the* ***snmp access*** *and* ***port*** *configurations in the* ***squid.conf*** *file.*

Security

Squid can use its role as an intermediary between web clients and a web server to implement access controls, determining who can access the web server and how. Squid does this by checking the access control lists (ACLs) of hosts and domains that have had controls placed on them. When it finds a web client from one of those hosts attempting to connect to the web server, it executes the control. Squid supports a number of controls with which it can deny or allow access to the web server by the remote host's web client (see Table 24-2). In effect, Squid sets up a firewall just for the web server.

The first step in configuring Squid security is to create ACLs. These are lists of hosts and domains for which you want to set up controls. You define ACLs using the **acl** command, creating a label for the systems on which you are setting controls. You then use commands such as **http_access** to define these controls. You can define a system, or a group of systems, by use of several **acl** options, such as the source IP address, the domain name, or even the time and date. For example, the **src** option is used to define a system or group of systems with a certain source address. To define a **mylan acl** entry for systems in a local network with the addresses 192.168.0.0 through 192.168.0.255, use the following ACL definition:

```
acl mylan src 192.168.0.0/255.255.255.0
```

Once it is defined, you can use an ACL definition in a Squid option to specify a control you want to place on those systems. For example, to allow access by the mylan group of local systems to the web through the proxy, use an **http_access** option with the **allow** action specifying **mylan** as the **acl** definition to use, as shown here:

```
http_access allow mylan
```

By defining ACLs and using them in Squid options, you can tailor your website with the kind of security you want. The following example allows access to the web through the proxy by only the **mylan** group of local systems, denying access to all others. Two **acl**

Option	Description
src *ip-address/netmask*	Client IP address
src *addr1-addr2/netmask*	Range of addresses
dst *ip-address/netmask*	Destination IP address
myip *ip-address/netmask*	Local socket IP address
srcdomain *domain*	Reverse lookup, client IP
dstdomain *domain*	Destination server from URL; for **dstdomain** and **dstdom_regex**, a reverse lookup is tried if an IP-based URL is used
srcdom_regex [-i] *expression*	Regular expression matching client name
dstdom_regex [-i] *expression*	Regular expression matching destination
time *[day-abbrevs] [h1:m1-h2:m2]*	Time as specified by day, hour, and minutes. Day abbreviations: S = Sunday, M = Monday, T = Tuesday, W = Wednesday, H = Thursday, F = Friday, A = Saturday
url_regex [-i] *expression*	Regular expression matching on whole URL
urlpath_regex [-i] *expression*	Regular expression matching on URL path
port *ports*	A specific port or range of ports
proto *protocol*	A specific protocol, such as HTTP or FTP
method *method*	Specific methods, such as GET and POST
browser [-i] *regexp*	Pattern match on user-agent header
ident *username*	String match on **ident** output
src_as *number*	Used for routing of requests to specific caches
dst_as *number*	Used for routing of requests to specific caches
proxy_auth *username*	List of valid usernames
snmp_community *string*	A community string to limit access to your SNMP agent

Table 24-2 Squid ACL Options

entries are set up: one for the local system and one for all others; **http_access** options first allow access to the local system and then deny access to all others.

```
acl mylan src 192.168.0.0/255.255.255.0
acl all src 0.0.0.0/0.0.0.0
http_access allow mylan
http_access deny all
```

The default entries that you will find in your **squid.conf** file, along with an entry for the mylan sample network, are shown here. You will find these entries in the ACCESS CONTROLS section of the **squid.conf** file.

```
acl all src 0.0.0.0/0.0.0.0
acl manager proto cache_object
acl localhost src 127.0.0.1/255.255.255.255
acl mylan src 192.168.0.0/255.255.255.0
acl SSL_ports port 443 563
```

The order of the **http_access** options is important. Squid starts from the first and works its way down, stopping at the first **http_access** option with an ACL entry that matches. In the preceding example, local systems that match the first **http_access** command are allowed, whereas others fall through to the second **http_access** command and are denied.

For systems using the proxy, you can also control what sites they can access. For a destination address, you create an **acl** entry with the **dst** qualifier. The **dst** qualifier takes as its argument the site address. Then you can create an **http_access** option to control access to that address. The following example denies access by anyone using the proxy to the destination site **rabbit.mytrek.com**. If you have a local network accessing the web through the proxy, you can use such commands to restrict access to certain sites.

```
acl myrabbit dst rabbit.mytrek.com
http_access deny myrabbit
```

The **http_access** entries already defined in the **squid.conf** file, along with an entry for the **mylan** network, are shown here. Access to outside users is denied, whereas access by hosts on the local network and the localhost (Squid server host) is allowed.

```
http_access allow localhost
http_access allow mylan
http_access deny all
```

You can also qualify addresses by domain. Often, websites can be referenced using only the domain. For example, a site called **mybeach.com** can be referenced using just the domain **mybeach.com**. To create an **acl** entry to reference a domain, use the **dstdomain** or **srcdomain** option for destination and source domains, respectively. Remember, such a reference refers to all hosts in that domain. An **acl** entry with the **dstdomain** option for **mybeach.com** restricts access to **mybeach.com**, **ftp.mybeach.com**, **surf.mybeach.com**, and so on. The following example restricts access to the **mybeach.com** site along with all other **.mybeach.com** sites and any hosts in the **mybeach.com** domain:

```
acl thebeach dstdomain .mybeach.com
http_access deny thebeach
```

You can list several domains or addresses in an **acl** entry to reference them as a group, but you cannot have one domain that is a subdomain of another. For example, if **mybeachblanket.com** is a subdomain of **mybeach.com**, you cannot list both in the same **acl** list. The following example restricts access to both **mybeach.com** and **mysurf.com**:

```
acl beaches dstdomain .mybeach.com .mysurf.com
http_access deny beaches
```

An **acl** entry can also use a pattern to specify certain addresses and domains. In the following example, access is denied to any URL with the pattern "chocolate" but allowed to all others:

```
acl Choc1 url_regex chocolate
http_access deny Choc1
http_access allow all
```

Squid also supports ident and proxy authentication methods to control user access. The following example allows only the users **dylan** and **chris** to use the Squid cache:

```
ident_lookup on
acl goodusers user chris dylan
http_access allow goodusers
http_access deny all
```

Caches

Squid primarily uses the Internet Cache Protocol (ICP) to communicate with other web caches. It also provides support for the more experimental Hypertext Cache Protocol (HTCP) and the Cache Array Routing Protocol (CARP).

Using the ICP protocols, your Squid cache can connect to other Squid caches or other cache servers, such as Microsoft proxy server, Netscape proxy server, and Novell BorderManager. This way, if your network's Squid cache does not have a copy of a requested web page, it can contact another cache to see if it is there instead of accessing the original site. You can configure Squid to connect to other Squid caches by connecting it to a cache hierarchy. Squid supports a hierarchy of caches denoted by the terms *child, sibling,* and *parent*. Sibling and child caches are accessible on the same level and are automatically queried whenever a request cannot be located in your own Squid's cache. If these queries fail, a parent cache is queried, which then searches its own child and sibling caches—or its own parent cache, if needed—and so on.

You can set up a cache hierarchy to connect to the main NLANR server by registering your cache using the following entries in your **squid.conf** file:

```
cache_announce 24
announce_to sd.cache.nlanr.net:3131
```

Connecting to Caches

Use **cache_peer** to set up parent, sibling, and child connections to other caches. This option has five fields. The first two consist of the hostname or IP address of the queried cache and the cache type (parent, child, or sibling). The third and fourth are the HTTP and the ICP ports of that cache, usually 3128 and 3130. The last is used for cache_peer options such as proxy-only to not save fetched objects locally, no-query for those caches that do not support ICP, and weight, which assigns priority to a parent cache. The following example sets up a connection to a parent cache:

```
cache_peer sd.cache.nlanr.net parent 3128 3130
```

Memory and Disk Configuration

Squid provides several options for configuring cache memory. The **cache_mem** option sets the memory allocated primarily for objects currently in use (objects in transit). If available, the space can also be used for frequently accessed objects (hot objects) and failed requests (negative-cache objects). The default is 8MB. The following example sets it to 256MB:

```
cache_mem 256 MB
```

You can further specify the minimum and maximum sizes of objects saved either on disk or in memory. On disk, you use **maximum_object_size** and **minimum_object_size**. The default maximum is 4KB. The default minimum is set to 0, indicating no minimum. For memory, you use **maximum_object_size_in_memory** and **minimum_object_size_in_memory**.

The **cache_swap_low** option lets you set bars for replacing objects in your cache.

To designate where cache objects are to be located, you use the **cache _dir** option. Here you specify what directories to use for your cache.

Administrative Settings

The email address for the administrator for your Squid cache is specified in the **cache_mgr** option.

If you run Squid as the root user, then Squid will change its user and group ID from **root** to **nobody**. The group ID will be changed to **nogroup**. This is to protect root user access. Should you run Squid as a user other than root, Squid will retain that original user as its user ID. If, when running Squid from the root user, you want to designate another user other than nobody, you can use **cache_effective_user** to change user IDs, and **cache_effective_group** to change the group.

You can also specify an special host name to be displayed in error messages. Use **visible_hostname** to set the name.

Logs

Squid keeps several logs detailing access, cache performance, and error messages.

- **access.log** holds requests sent to your proxy.
- **cache.log** holds Squid server messages such as errors and startup messages.
- **store.log** holds information about the Squid cache such as objects added or removed.

You can use the cache manager (**cachemgr.cgi**) to manage the cache and view statistics on the cache manager as it runs. To run the cache manager, use your browser to execute the **cachemgr.cgi** script (this script should be placed in your web server's **cgi-bin** directory).

Web Server Acceleration: Reverse Proxy Cache

Though Squid caches can enhance access by clients to a web server, Squid can also reduce the load on a web server. Web servers that become overwhelmed by requests can move their cacheable pages to a Squid proxy server that can serve as a kind of alternate site, handling

requests for those pages. In effect, the web server becomes accelerated. Such a cache is known as a reverse proxy cache, focusing on the server instead of the client. A reverse proxy cache will intercept requests to a server, processing any for its cached pages. Only requests for noncached pages are forwarded to the original web server.

To configure a reverse proxy cache, you use the **http_port** directive with the **accel** option (for Squid 2.4 and earlier you use **httpd_accel** directives). To specify a particular location for the accelerators, you use **defaultsite** (this replaces httpd_accel_host used in Squid 2.4 and earlier).

```
http_port 3128
http_port 192.168.0.25:80 accel defaultsite=rabbit.mytrek.com
```

If your Squid proxy server and the web server are operating on the same host, you need to specify the port that the web server is using. This cannot be the same port that Squid is using. You use the **cache_peer** directive with the **port** option to specify the server port. You then specify the address of the web server with the **cache_peer** directive and the **originserver** option. In the following example, the web server is using port 80, whereas Squid is using port 3128:

```
http_port 3128 # Port of Squid proxy
cache_peer port 80
cache_peer originserver localhost # IP address of web server
```

With Squid 2.6, transparent caches are supported directly as an **http_port** option, **transparent**.

```
http_port 3218 transparent
```

To specify the use of host directives (virtual hosts), you use the **vhost** option. This replaces the **httpd_accel_uses_host_header** directive used in earlier versions. For virtual IP hosts you use **vport**.

```
http_port vhost
```

In addition, DNS entries for the external network would use the IP address of the proxy server for the web server host name, directing all the web server requests to the proxy server. DNS entries for the internal network would use the web server's IP address for the web server host name, allowing the proxy to redirect noncached requests on to the web server. If your network uses only one DNS server, you can set up a Split DNS server to specify internal and external addresses.

This page intentionally left blank

25 CHAPTER

Mail Servers

Mail servers provide Internet users with electronic mail services. They have their own TCP/IP protocols such as the Simple Mail Transfer Protocol (SMTP), the Post Office Protocol (POP), and the Internet Mail Access Protocol (IMAP). Messages are sent across the Internet through mail servers that service local domains. A *domain* can be seen as a subnet of the larger Internet, with its own server to handle mail messages sent from or received for users on that subnet. When a user mails a message, it is first sent from his or her host system to the mail server. The mail server then sends the message to another mail server on the Internet, the one servicing the subnet on which the recipient user is located. The receiving mail server then sends the message to the recipient's host system.

At each stage, a different type of operation takes place using different agents (programs). A mail user agent (MUA) is a mail client program, such as mail or Elm. With an MUA, a user composes a mail message and sends it. Then a mail transfer agent (MTA) transports the messages over the Internet. MTAs are mail servers that use SMTP to send messages across the Internet from one mail server to another, transporting them among subnets. On Linux and Unix systems, the commonly used MTA is Sendmail, a mail server daemon that constantly checks for incoming messages from other mail servers and sends outgoing messages to appropriate servers. Other MTAs becoming more popular are Postfix, Exim, Courier, and Qmail (see Table 25-1). Incoming messages received by a mail server are distributed to a user by mail delivery agents (MDAs). Most Linux systems use procmail as their MDA, taking messages received by the mail server and delivering them to user accounts (see **procmail.org** for more information).

Mail Transport Agents

Many Linux distributions will automatically install and configure either Sendmail or Postfix for you. On starting your system, you can send and receive messages between local users using Sendmail or Postfix. You can also set up your Linux system to run a POP server. POP servers hold users' mail until they log in to access their messages, instead of having mail sent to their hosts directly. Both Postfix and Sendmail will be discussed in this chapter.

Courier is a fast, small, and secure MTA that maintains some compatibility with Sendmail. The Courier software package also includes POP, IMAP, and webmail servers along with mailing list services. It supports extensive authentication methods including shadow passwords, PAM, and LDAP.

Agent	Description
Sendmail	Sendmail mail transfer agent, supported by the Sendmail consortium **sendmail.org**
Postfix	Fast, easy-to-configure, and secure mail transfer agent compatible with Sendmail and designed to replace it **postfix.org**
Qmail	Fast, flexible, and secure MTA with its own implementation and competitive with Postfix **qmail.org**
Exim	MTA based on smail3 **exim.org**
Courier	Courier MTA **courier-mta.org**

TABLE 25-1 Mail Transfer Agents

Exim is a fast and flexible MTA similar to Sendmail. Developed at the University of Cambridge, it has a very different implementation than Sendmail.

Qmail is also a fast and secure MTA, but it has little compatibility with Sendmail. It has its own configuration and maintenance files. Like Postfix, it has a modular design, using a different program for each mail task. It also focuses on security, speed, and easy configuration.

NOTE *Messages sent within a single standalone system require a loopback interface. Most Linux distributions do this automatically for you during the installation process. A* loopback interface *enables your system to address itself, allowing it to send and receive mail to and from itself. A loopback interface uses the hostname* ***localhost*** *and a special IP address reserved for use by local systems, 127.0.0.1. You can examine your* ***/etc/hosts*** *file to see if your loopback interface has been configured as the localhost. You should see* ***127.0.0.1 localhost*** *listed as the first entry.*

Received Mail: MX Records

A mail address consists of a username and a host address. The host address takes the form of a fully qualified domain name, listing the hostname and the domain name, separated by periods. Most uses of a hostname, such as FTP connections, translate the hostname into an IP address and use the IP address to locate the host system. Mail messages operate nearly the same way. However, they make use of the Domain Name Service to determine which host to actually send a message to. The host specified in the mail address may not be the host to which delivery should actually be made. Different networks will often specify a mail server to which mail for the hosts in a network should be delivered. For example, mail addressed to the **rabbit.mytrek.com** host may actually be delivered to the **turtle.mytrek.com** host. **turtle.mytrek.com** may be running a POP mail server that users on **rabbit.mytrek.com** can access to read their mail.

Such mail servers are associated with different hosts by mail exchange records, known as MX records, in a network's DNS configuration. When mail is received in a network, the

network's DNS configuration is first checked for MX records to determine if the mail is to be delivered to a host different from that in the mail message address. For example, the following MX record says that any mail for the **rabbit.mytrek.com** host is to be delivered to the **turtle.mytrek.com** host; **turtle.mytrek.com** is the mail exchanger for **rabbit.mytrek.com**:

```
rabbit.mytrek.com. IN   MX    0   turtle.mytrek.com.
```

A host can have several mail exchangers, each with a different priority. If one is down, the one with next highest priority will be accessed. Such a design provides for more robust mail delivery, letting a few well-maintained servers handle received mail, instead of each host on its own.

Mail exchange records are also used for mail addresses for which there are no hosts. For example, you can designate virtual hosts or use the domain name as an address. To use a domain name, you map an MX record with the domain name to a mail server on the network. Mail addressed to the domain name is sent to the mail server. For example, with the following MX record, mail sent to **mytrek.com** would be delivered to **turtle.mytrek.com**, which would be running a mail server like Sendmail:

```
mytrek.com. IN   MX     0   turtle.mytrek.com.
```

Mail addressed to george@mytrek.com would be sent to george@turtle.mytrek.com.

NOTE *MX records are used not only for incoming mail, but also for outgoing mail. An MX record can specify a mail server to use for relaying mail from a given host out to a larger network.*

MX records come into play with certain mail configurations such as masquerading or centralized mail services. MX records are not required. If you have a standalone system or a small network with only a few hosts, you may want mail received directly by different hosts.

Postfix

Postfix is a fast, secure, and flexible MTA designed to replace Sendmail while maintaining as much compatibility as possible. Written by Wietse Venema and originally released as the IBM Secure Mailer, it is now available under the GNU license (**postfix.org**). Postfix was created with security in mind, treating all incoming mail as potential security risks. Postfix uses many of the same Sendmail directories and files and makes use of Sendmail wrappers, letting Sendmail clients interact seamlessly with Postfix servers. Postfix is also easier than Sendmail to configure, using its own configuration file.

Instead of one large program, Postfix is implemented as a collection of smaller programs, each designed to perform a specific mail-related task. A Postfix master daemon runs continuously and manages the use of the other Postfix daemons, running them only as needed. A **bounce** daemon handles undeliverable mail, a **trivial-rewrite** daemon redirects messages, and the **showq** daemon provides information on the print queues.

Postfix Commands

Several Postfix commands allow you to manage your server tasks. The **`sendmail`** command sends messages. You use **`mailq`** to display the status of your mail queues. The **`newaliases`**

command takes mail aliases listed in the aliases files and stores them in a database file that can be used by Postfix.

The **postmap** command is used to maintain various database files used by Postfix, such as the alias file for mail aliases and the access file that restricts messages received by the server. You can also implement these database files as SQL databases like MySQL, allowing for easier management. The **mysql_table** Man page provides detailed information on how to configure SQL database support (check **pgsql_table** for PostgreSQL database support). You could also use LDAP instead of SQL (**ldap_table**).

In addition, Postfix provides lower-level tools, all beginning with the term **post**, such as the **postalias** command, which maintains the alias database, and **postcat**, which displays print queue files.

Postfix Configuration: main.cf

Postfix configuration is handled by setting parameters in its configuration file, **main.cf**. A default **/etc/postfix/main.cf** file is installed with Postfix, with most of the essential configuration values already set. Parameter names tend to be user friendly. For example, directory locations are specified by parameters ending in the term **directory**, such as **queue_directory** for the location of Postfix queues and **daemon_directory** for the location of the Postfix daemons. Defaults are already implemented for most parameters. For example, defaults are set for particular resource controls, such as message size, time limits, and the number of allowed messages per queue. You can edit the **main.cf** file to change the parameter values to meet your own needs. After making any changes, you need only to reload the configuration using the **postfix reload** command:

```
postfix reload
```

Network Parameters

You will most likely need to set several network parameters. To ease this process, Postfix defines parameters that hold key network information, such as **myhostname**, which holds the hostname of your system, and **mydomain**, which holds the domain name of your network. For example, **myhostname** would be set to the host **turtle.mytrek.com**, whereas **mydomain** would be just **mytrek.com**. Parameters like **myhostname** and **mydomain** are themselves used as values assigned to other parameters. In the next example, **myhostname** and **mydomain** are set to the host the mail server is running on and its network domain:

```
myhostname=turtle.mytrek.com
mydomain=mytrek.com
```

The **myorigin** parameter specifies the origin address for email sent by the server. By default, this is set to the value of the parameter **myhostname**, as shown here. Note that a **$** precedes the **myhostname** variable to evaluate it.

```
myorigin=$myhostname
```

If you are using a single system directly attached to the Internet, you may want to keep this configuration, labeling mail as being sent by your host. However, if your system is operating as a gateway for a network, your mail server is sending out mail from different

hosts on that network. You may wish to change the origin address to the domain name, so that mail is perceived as sent from the domain.

```
myorigin=$mydomain
```

Local Networks

The **mydestination** parameter holds the list of domains that your mail server will receive mail for. By default, these include **localhost** and your system's hostname.

```
mydestination = $myhostname localhost.$mydomain
```

If you want the mail server to receive mail for an entire local network, you need to also specify its domain name. That way, the server can receive mail addressed just to the domain, instead of your specific host.

```
mydestination = $myhostname localhost.$mydomain $mydomain
```

Also, if your host goes by other hostnames and there are DNS records identifying your host by those names, you need to specify those names as well. For example, your host could also be a web server to which mail could be directed. A host **turtle.mytrek.com** may also be identified as the website **mytrek.com**. Both names would have to be listed in the **mydestination** parameter.

```
mydestination = $myhostname localhost.$mydomain $mydomain www.$mydomain
```

If your system is a gateway for one or more local networks, you can specify them with the **mynetworks** parameter. This allows your mail server to relay mail addressed to those networks. Networks are specified using their IP addresses. The **relay_domains** parameter lets you specify domain addresses of networks for which you can relay messages. By default, this is set to **mydestination**:

```
mynetworks=192.168.0.0
relay_domains=$mydestination
```

Hosts within the local network connected to the Internet by a gateway need to know the identity of the relay host, the mail server. You set this with the **relayhost** parameter. Also, **myorigin** should be set to just **mydomain**. If there is a DNS server identifying the gateway as the mail server, you can just set **relayhost** to the value of **mydomain**. If not, then **relayhost** should be set to the specific hostname of the gateway/mail server. If your local network is not running a DNS server, be sure to set **disable_dns_lookups** to **yes**.

```
relay_host=$mydomain
```

Direct Connections

If your system is directly connected to the Internet and you use an ISP (Internet service provider) for receiving mail, you can configure Postfix as a null client to only send mail. Set the **relay_host** parameter to just your own domain name. Also, in the **master.cf** file, comment out the SMTP server and local delivery agent entries.

```
relayhost = $mydomain
```

Masquerading

If your mail server is operating on a gateway for a local network and you want to hide the hosts in that network, you can opt to masquerade the localhost, letting it appear that all mail is coming from the domain in general, instead of a particular host. To set this option, you use the **masquerade_domains** parameter. In the following example, all mail sent by a localhost such as **rabbit.mytrek.com** will be addressed as coming from **mytrek.com**. Thus a message sent by the user **chris@rabbit.mytrek.com** is sent out as coming from **chris@mytrek.com**:

```
masquerade_domains = $mydomain
```

Received mail is not masqueraded by default. This allows Postfix to still deliver received mail to particular hosts. If you want received mail to also be masqueraded, you have to add the **envelope_recipients** parameter to the list of values assigned to the **masquerade_class** parameter. In that case, Postfix will no longer be able to deliver received mail.

Virtual Domains and Virtual Accounts

If your network has implemented virtual domains, you will need to set up a virtual domain table and then specify that table with the **virtual_maps** option. Setting up a table is a simple matter of listing virtual names and their real addresses in a text file such as **/etc/postfix/virtual**. Then use the **postmap** command to create a Postfix table:

```
postmap /etc/postfix/virtual
```

In the **main.cf** file, specify the table with the **virtual_maps** parameter. Postfix will then use this table to look up virtual domains.

```
virtual_maps = hash:/etc/postfix/virtual
```

NOTE *See the Postfix FAQ at **postfix.org** for detailed information on how to set up Postfix for a gateway, a local workstation, or a host directly connected to the Internet (null server).*

Instead of using mail accounts for actual users on a system, you can set up a virtual account. Virtual accounts can be managed either in standard Postfix text files, in SQL databases, or as LDAP entries. SQL databases are preferred for managing a large number of virtual accounts. For SQL support, you first create tables in a MySQL database for domains (the virtual domains), users (user accounts), and forwarding (aliases). Corresponding virtual domain configuration files will list information like the database, tables, and host to use, such as a **mysql_virt.cf** for SQL database access and **mysql_users.cf** for accessing the user table. Check the documentation at **postfix.org** for detailed information.

Postfix Greylisting Policy Server

Postfix also supports greylisting with the Postfix Greylisting Policy Server. Greylisting blocks spammers based on their mailing methods rather than content, relying on the fact that spammers will not attempt retries if rejected (**greylisting.org**). Messages from new, previously unknown sources are rejected, whereupon a valid MTA will retry, whereas a spammer will not. To support the Greylisting Policy Server, Postfix is configured to delegate

policy access to a server. In the **/etc/postfix** directory you can use the postgrey_whitelist files to exclude email addresses from greylisting.

The Greylisting Policy Server is run as a standalone server, using its own startup script. The postgrey Man page provides detailed information about the server's options.

Controlling User and Host Access

With an access file, you can control access by certain users, hosts, and domains. The access file works much like the one used for Sendmail. Entries are made in a text file beginning with the user, host, or domain name or address, followed by an action to take. A user, host, or domain can be accepted, rejected with no message, or rejected with a message. Once entries are made, they can be installed in a Postfix database file with the **postmap** command:

```
postmap /etc/postfix/access
```

You can then use the access file in various Postfix operations to control clients, recipients, and senders.

Access can also be controlled by use of the Mail Abuse Prevention System (MAPS), which provides the RBL+ service, a collection of mail address DNS-based databases (**mail-abuse.com**). These databases, like the Realtime Blackhole List (RBL), list mail addresses that are known to be used by mail abusers. A domain or host is matched against a list maintained by the service, which can be accessed on a local server or directly from an online site. Various Postfix operations let you use MAPS databases to control access by clients, recipients, or senders.

Header and Body Checks

With the **header_checks** parameter, you can specify a Postfix table where you can list criteria for rejecting messages. Check the **/etc/postfix/header_checks** file for details. The criteria are patterns that can match message headers. You can have matching messages rejected, rejected with a reply, simply deleted, or logged with a warning. You have the option of taking several actions, including REJECT, DISCARD, WARN, HOLD, and IGNORE.

```
header_checks = regexp:/etc/postfix/header_checks
```

The database, in this case **/etc/postfix/header_checks**, will have lines, each with a regular expression and a corresponding action. The regular expression can either be a standard regular expression as denoted by **regexp** in the **header_checks** parameter, or conform to a Perl Compatible Regular Expression, **prece**.

The **body_checks** parameter lets you check the body of text messages, line by line, using regular expressions and actions like those used for **header_checks** in an **/etc/postfix/body_checks** file.

Controlling Client, Senders, and Recipients

With the **smtpd_client_restrictions** parameter, you can restrict access to the mail server by certain clients. Restrictions you can apply include **reject_unknown_client_hostname**, which will reject any clients with unresolved addresses; **permit_mynetworks**, which allows access by any clients defined by **mynetworks**; and **check_client_access**, which will check

an access database to see if a client should be accepted or rejected. The **reject_rbl_client** and **reject_rhsbl_client** parameters will reject clients from specified domains.

```
smtpd_client_restrictions = permit_mynetworks, \
             reject_unknown_client, check_client_access, reject_maps_rbl
```

The **reject_rbl_client** restriction rejects domain addresses according to a specified MAPS service. The site can be an online site or a local one set up to provide the service. The **reject_rhsbl_client** restriction rejects host addresses.

```
smtpd_client_restrictions = reject_rbl_client relays.mail-abuse.org
```

To implement restrictions from an access file, you can use the **hash** directive and the name of the file.

```
smtpd_client_restrictions = hash:/etc/postfix/access
```

The corresponding **smtpd_sender_restrictions** parameter works much the same way as its client counterpart but controls access from specific senders. It has many of the same restrictions but adds **reject_non_fqdn_sender**, which will reject any mail header without a fully qualified domain name, and **reject_sender_login_mismatch**, which will require sender verification. The **reject_rhsbl_sender** restriction rejects domain addresses according to a specified MAPS service.

The **smtpd_recipient_restrictions** parameter will restrict the recipients the server will accept mail for. Restrictions include **permit_auth_destination**, which allows authorized messages, and **reject_unauth_destination**, which rejects unauthorized messages. The **check_recipient_access** restriction checks local networks for a recipient address. The **reject_unknown_recipient_domain** restriction rejects recipient addresses with no DNS entry. The **reject_rhsbl_recipient** restriction rejects domain addresses according to a specified MAPS service.

You can further refine restrictions with parameters such as **smtpd_helo_restrictions**, which requires a HELO command from a client. Restriction parameters include **reject_invalid_hostname**, which checks for faulty syntax, **reject_unknown_hostname**, for hosts with no DNS entry, and **reject_non_fqdn_hostname** for hosts whose names are not fully qualified. The **strict_rfc821_envelopes** parameter will implement strict envelope protocol compliance.

Sendmail

Sendmail operates as a server to both receive and send mail messages. Sendmail listens for any mail messages received from other hosts and addressed to users on the network hosts it serves and, at the same time, handles messages users are sending out to remote users, determining what hosts to send them to. You can learn more about Sendmail at **sendmail.org**, including online documentation and current software packages. The Sendmail newsgroup is **comp.mail.sendmail**. You can also obtain a commercial version from **sendmail.com**.

The domain name server for your network designates the host that runs the Sendmail server. This is your mail host. Messages are sent to this host, whose Sendmail server then sends the message to the appropriate user and its host. In your domain name server configuration file, the mail host entry is specified with an MX entry. To print the mail queue of messages for future delivery, you can use **mailq** (or **sendmail -v -q**). This runs Sendmail with instructions to print the mail queue.

The Sendmail software package contains several utilities for managing your Sendmail server. These include mailq, which displays the queue of outgoing messages; mailstats, which shows statistics on mail server use; hoststat, which provides the stats of remote hosts that have connected with the mail server; and praliases, which prints out the mail aliases listed in the **/etc/aliases** file. Some utilities, like mailq and hoststat, simply invoke Sendmail with certain options. Others, like mailstats and praliases, are separate programs.

Sendmail now maintains all configuration and database files in the **/etc/mail** directory. Here you will find the Sendmail macro configuration file, **sendmail.mc**, as well as several database files (see Table 25-2). Many have changed their names with the release of Sendmail 8.10. For example, the help file is now **/etc/mail/helpfile** instead of **/etc/sendmail.ht**. Specialized files provide support for certain features such as **access**, which lets you control access by different hosts and networks to your mail server, and virtusertable, which lets you designate virtual hosts. These files have both text and database versions. The database version ends with the extension **.db** and is the file actually used by Sendmail. You make your entries in the text version and then effect the changes by generating a corresponding database version. Database versions are generated using the **makemap** command with the **hash** option and a redirection operation for the text and database file. For example, to deny access to a particular host, you place the appropriate entry for it in the **/etc/mail/access** file, editing the file using any text word processor. Then, to generate the **/etc/mail/access.db** version of the access file, you change to the **/etc/mail** directory and use the following command:

```
cd /etc/mail
makemap hash access < access
```

To regenerate all the database files, just use the **make** command in the **/etc/mail** directory:

```
make
```

Certain files and directories are used to manage the mail received and sent. Incoming mail is usually kept in the **/var/spool/mail** directory, and outgoing messages are held in the **/var/spool/mqueue** directory, with subdirectories for different users. Monitoring and error messages are logged in the **/var/log/maillog** file.

NOTE *If your mail server services several hosts, you will need to enter them in the* ***/etc/mail/local-host-names*** *file.*

Aliases and LDAP

Sendmail can now support the Lightweight Directory Access Protocol (LDAP), which enables the use of a separate server to manage Sendmail queries about user mail addresses. Instead of maintaining aliases and **virtusertable** files on different servers, Sendmail uses LDAP support to simply use one centralized LDAP server to locate recipients. Mail addresses are looked up in the LDAP server, instead of having to search several aliases and **virtusertable** files on different servers. LDAP also provides secure authentication of users, allowing controlled access to mail accounts. The following example enables LDAP support on Sendmail in the **sendmail.mc** file:

```
FEATURE('ldap_routing')dnl
LDAPROUTE_DOMAIN('mytrek.com')dnl
```

File	Description
/etc/mail/sendmail.cf	Sendmail configuration file
/etc/mail/sendmail.mc	Sendmail M4 macro configuration file
/etc/mail/submit.cf	Sendmail configuration file for mail submission mode where Sendmail does not run as a server but merely submits mail
/etc/mail/submit.mc	Sendmail M4 macro configuration file for Sendmail mail submission mode
/etc/aliases	Sendmail aliases file for mailing lists
/etc/aliases.db	Sendmail aliases database file generated by the **`newaliases`** command using the aliases file
/etc/mail/access	Sendmail access text file; access control for screening or relaying messages from different hosts, networks, or users; used to generate the **access.db** file
/etc/mail/access.db	Sendmail access database file, generated from the access text file
/etc/mail/local-host-names	Sendmail localhosts file for multiple hosts using the same mail server (formerly **sendmail.cw**)
/etc/mail/trusted-users	Sendmail trusted users file (formerly **sendmail.ct**)
/etc/mail/error-header	Sendmail error header file (formerly **sendmail.oE**)
/etc/mail/helpfile	Sendmail help file (formerly **sendmail.ht**)
/etc/mail/statistics	Sendmail statistics file (formerly **sendmail.st**)
/etc/mail/virtusertable	Sendmail virtual user table text file; maps user virtual domain addresses, allowing virtual domains to be hosted on one system; make entries in this file and then use it to generate the **virtusertable.db** file
/etc/mail/virtusertable.db	Sendmail virtual user table database generated from the **virtusertable** file
/etc/mail/mailertable	Sendmail mailer table text file used to override routing for your domains
/etc/mail/mailertable.db	Sendmail mailer table database file, generated from the **mailertable** file
/etc/mail/userdb	Sendmail user database file
/etc/mail/domaintable	Sendmail **domaintable** file, maps a domain name to another domain name
/etc/mail/domaintable.db	Sendmail **domaintable** database file, generated from the **domaintable** file
/var/spool/mail	Incoming mail
/var/spool/mqueue	Outgoing mail
/var/spool/maillog	Mail log file

TABLE 25-2 Sendmail Files and Directories

Alternatively, Sendmail still supports the use of aliases, for either sent or received mail. It checks an aliases database file called **aliases.db** that holds alias names and their associated email addresses. This is often used for administrator mail, where mail may be sent to the system's root user and then redirected to the mail address of the actual system administrator. You can also alias host addresses, enabling you to address hosts on your network using only their aliases. Alias entries are kept in the **/etc/aliases** file. This file consists of one-line alias records associating aliases with user addresses. You can edit this file to add new entries or to change old ones. They are then stored for lookup in the **aliases.db** file using the command **newaliases**, which runs Sendmail with instructions to update the **aliases.db** file.

Aliases allow you to give different names for an email address or collection of email addresses. One of its most useful features is to create a mailing list of users. Mail addresses to an alias will be sent to the user or list of users associated with the alias. An alias entry consists of an alias name terminated by a colon and followed by a username or a comma-separated list of users. For example, to alias **filmcritic** with the user **george@rabbit.mytrek.com**, you use the following entry:

```
filmcritic:   george@rabbit.mytrek.com
```

To alias **singers** with the local users **aleina** and **larisa**, you use

```
singers:     aleina, larisa
```

You can also use aliases as the target addresses, in which case they will expand to their respective user addresses. For example, the **performers** alias will expand through the **filmcritic** and **singers** aliases to the users **george@rabbit.mytrek.com**, **aleina**, and **larisa**:

```
performers:     filmcritic, singers
```

Once you have made your entries in the **/etc/mail/aliases** file, you need to generate a database version using the **newaliases** command:

```
newaliases
```

NOTE *Sendmail now supports a mail submission mode in which Sendmail does not run as a server along with the corresponding root privileges. Instead, it merely submits mail. In this mode, Sendmail can be invoked directly by email clients to send mail. The mail submission mode uses the* ***/etc/mail/submit.cf*** *configuration file, which can be configured using macros in the* ***/etc/mail.submit.mc*** *file.*

Sendmail Configuration

The main Sendmail configuration file is **sendmail.cf**, located in the **/etc** directory. This file consists of a sometimes lengthy list of mail definitions that set general options, designate MTAs, and define the address rewrite rules. A series of options sets features, such as the maximum size of mail messages or the names of host files. The MTAs are those mailers through which Sendmail routes messages. The rewrite rules "rewrite" a mail address to route a message through the appropriate Internet connections to its destination (these rules

can be complex). Check the Sendmail HOWTO and the online documentation for a detailed explanation.

The **sendmail.cf** definitions can be complex and confusing. To simplify the configuration process, Sendmail supports the use of macros you can use to generate the **sendmail.cf** file using the M4 preprocessor (this requires installation of the **sendmail-cf** package). Macros are placed in the **/etc/mail/sendmail.mc** file. Here, you can use macros to designate the definitions and features you want for Sendmail, and then the macros are used to generate the appropriate definitions and rewrite rules in the **sendmail.cf** file. As part of the Sendmail package, several specialized versions of the **sendmail.mc** file are made available in the **/usr/share/sendmail-cf** directory. These begin with a system name and have the suffix **.mc**. On many distributions, a specialized version tailored to your distribution is already installed as your **/etc/mail/sendmail.mc** file.

Once you configure your **sendmail.mc** file, you use the following command, run in the **/etc/mail** directory, to generate a **sendmail.cf** file (be sure to first back up your original **sendmail.cf** file). You can rename the **sendmail.mc** file to reflect the specific configuration, and you can have as many different **.mc** files as you want and use them to implement different configurations. :

```
make -C /etc/mail
```

Alternatively, you can use the original **m4** macro command:

```
m4 sendmail.mc > /etc/mail/sendmail.cf
```

You will then need to restart the Sendmail server to make the configuration effective.

In the **sendmail.mc** file, you configure different aspects of Sendmail using either a **define** command to set the value of Sendmail variables, or a Sendmail macro that has already been defined to set a particular Sendmail feature. For example, to assign the **PROCMAIL_PATH** variable to the directory **/usr/bin/procmail**, you use the following:

```
define('PROCMAIL_MAILER_PATH','/usr/bin/procmail')
```

Similarly, if there are variables that you do not want defined, you can remove them with the **undefine** command:

```
undefine('UUCP_RELAY')
```

To specify the type of operating system that your Sendmail server is running on, you use the **OSTYPE** Sendmail macro. The following example specifies the Linux operating system:

```
OSTYPE('linux')
```

The **MAILER** macro specifies the mail delivery agents (MDAs) to be used. You may have more than one. Usually, you will need a mail delivery agent such as procmail for delivering mail to hosts on your network. In addition, Sendmail in effect operates as an MDA to receive messages from hosts in its local network, which it will then send out to the larger network.

```
MAILER(procmail)
MAILER(smtp)
```

Sendmail also supports an extensive number of features that you need to explicitly turn on. You can do this with the Sendmail **FEATURE** macro. See Table 25-3 for a list of commonly used Sendmail features. The following example turns on the **redirect** feature, which is used to inform a sender that a recipient is now at a different address:

```
FEATURE(redirect)
```

Feature	Description
use_cw_file	Checks for hosts served by the mail server **/etc/mail/local-host-names** file.
use_ct_file	Reads a list of users from the **/etc/trusted-users** file. These are trusted users that can change the sender name for their messages.
redirect	Rejects all mail addressed to address REDIRECT, providing a forwarding address is placed in the **/etc/aliases** file.
mailertable	Uses a mailer table file, **/etc/mail/mailertable**, to override routing for particular domains.
domaintable	Uses a domain table file, **/etc/mail/domaintable,** to map one domain to another. Useful if you change your domain name.
allmasquerade	Causes recipient addresses to also masquerade as being from the masquerade host.
masquerade_entire_domain	Masquerades all hosts within the domain specified in **MASQUERADE_AS**.
masquerade_envelope	Masquerades envelope sender and recipient along with headers.
virtusertable	For virtual hosts; maps virtual addresses to real addresses.
nullclient	Turns a Sendmail server into a null client, which simply forwards mail messages to a central mail server for processing.
local_procmail	Uses procmail as the local mailer.
smrsh	Uses the Sendmail Restricted Shell (smrsh) for mailing.
promiscuous_relay	Allows you to relay mail, allowing mail to be received from outside your domain and sent on to hosts outside your domain.
relay_entire_domain	Allows any host in your domain to relay mail (default limits this to hosts in the access database).
relay_hosts_only	Checks for relay permission for particular hosts instead of domains.

TABLE 25-3 Sendmail Features

Feature	Description
accept_unqualified_senders	Allows sender email addresses to be single usernames instead of just fully qualified names that include domain names.
accept_unresolvable_domains	Allows Sendmail to accept unresolvable domain names. Useful for those users in a local network blocked by a firewall from the full DNS namespace. By default, Sendmail requires domains in addresses to be resolvable with DNS.
access_db	Accepts or rejects mail from domains and hosts in the access database.
blacklist_recipients	Blocks mail to certain users, such as those that should never receive mail—like the users **nobody** and **host**.
dnsbl	Rejects hosts in the Realtime Blackhole List. Managed by MAPS (Mail Abuse Prevention System LLC) and designed to limit transport of unwanted mass email (**mail-abuse.org**).
ldap_routing	Enables LDAP use.

TABLE 25-3 Sendmail Features (*continued*)

In addition, you can set certain configuration options. These are variables beginning with the prefix **conf** that you can set and assign values to using the **define** command. There are an extensive number of configuration options, most of which you will not need to change. The following example defines the **confAUTO_REBUILD** configuration option, which will automatically rebuild the aliases database, if needed:

```
define('confAUTO_REBUILD')
```

Certain macros and types of macros need to be placed in the **sendmail.mc** file in a particular sequence, as shown here. Notice that **MAILER** is toward the end and **OSTYPE** at the beginning. Local macro definitions (**define**) and **FEATURE** entries follow the **OSTYPE** and **DOMAIN** entries:

```
VERSIONID
OSTYPE
DOMAIN
define
FEATURE
local macro definitions
MAILER
LOCAL_RULE_*
LOCAL_RULESETS
```

The local macro and configuration option definitions that affect a particular feature need to be entered before the **FEATURE** entry. For example, the **redirect** feature uses the aliases file. Any local definition of the aliases file needs to be entered before the **redirect** feature.

```
define('ALIAS_FILE','/etc/aliases')
FEATURE(redirect)
```

You need to be careful how you enter comments into a **sendmail.mc** file. This file is read as a stream of macros, ignoring all white spaces, including newlines. No special comment characters are looked for. Instead, you have to simulate comment indicators using the **dnl** or **divert** commands. The **dnl** command instructs that all characters following that **dnl** command up to and including the next newline are to be ignored. If you place a **dnl** command at the beginning of a text line in the **sendmail.mc** file, it has the effect of turning that line into a comment, ignoring everything on that line—including its newline. Even empty lines will require a **dnl** entry to ignore the newline character:

```
dnl you will have to /etc/mail/sendmail.cf by running this the m4
dnl macro config through preprocessor:
dnl
```

Alternatively, you can use the **divert** command. The **divert** command will ignore all data until another **divert** command is reached:

```
divert(-1)
 This is the macro config file used to generate
 the /etc/mail/sendmail.cf file. If you modify the file regenerate
 you will have to regenerate /etc/mail/sendmail.cf by running the m4
 macro
divert(0)
```

For Sendmail to work at all, it requires only that the **OSTYPE** and **MAILERS** macros be defined, along with any needed features and options. A very simple Sendmail file is shown here:

mysendmail.mc

```
dnl My sendmail.mc file
OSTYPE('linux')
define('PROCMAIL_MAILER_PATH','/usr/bin/procmail')
FEATURE(redirect)
MAILER(procmail)
MAILER(smtp)
```

A **sendmail.mc** file usually contains many more entries, particularly for parameters and features. Check the **/etc/mail/sendmail.mc** file on your system to see the standard default entries for Sendmail.

Sendmail Masquerading

For a mail server that is relaying messages from localhosts to the Internet, you may want to masquerade the source of the messages. In large networks that have their own mail servers connected to the Internet, Sendmail masquerading can make messages sent by localhosts appear to be sent by the mail server. Their host address will be replaced by the mail server's address. Returned mail can then be sent to the mail server and held in POP or IMAP server mailboxes that can be later accessed by users on the localhosts. Also, entries in the server's virtual user table could forward mail to corresponding users in localhosts.

Masquerading is often used to mask localhosts with a domain name. Any subdomains can also be masqueraded. This method can be applied to situations where an ISP or your network administrator has assigned your network its own domain name. You can then mask all mail messages as coming from your domain name instead of from particular hosts or from any subdomains you may have. For example, if a network's official domain name is **mytrek.com**, all messages from the hosts in the **mytrek.com** network, such as **rabbit.mytrek.com** and **turtle.mytrek.com**, can be masqueraded to appear as just coming from **mytrek.com**. Should the **mytrek.com** network have a subnetwork whose domain is **mybeach.com**, any messages from **mybeach.com** can also be masqueraded as coming from **mytrek.com**.

Masquerading is turned on with the **MASQUERADE_AS** command. This takes as its argument the name you want to masquerade your mail as. Normally the name used is just the domain name, without the mail host. In the following example, the mail is masqueraded as simply **mytrek.com**. Mail sent from a localhost like **turtle.mytrek.com** will appear to be sent by just **mytrek.com**:

```
MASQUERADE_AS('mytrek.com')dnl
```

You will also have to specify the hosts and domains on your local network that your Sendmail server should masquerade. If you have decided to masquerade all the hosts in your local network, you need to set the **masquerade_entire_domain** feature, as in

```
FEATURE('masquerade_entire_domain')dnl
```

If, instead, you want to masquerade particular hosts or your domain has several subdomains that you want masqueraded, you list them in the **MASQUERADE_DOMAIN** entry. You can list either particular hosts or entire domains. For example, given a local network with the localhosts **turtle.mytrek.com** and **rabbit.mytrek.com**, you can list them with the **MASQUERADE_DOMAIN** to have them masqueraded. The domain they are masqueraded as is specified in the **MASQUERADE_AS** entry.

```
MASQUERADE_DOMAIN('turtle.mytrek.com rabbit.mytrek.com')dnl
```

If you want to masquerade all the hosts in your local network, you can simply list your local network's domain name. If your local network also supports several subdomains, you can list those as well to masquerade them. For example, to masquerade all the hosts in the **mybeach.com** domain, you use the following entry:

```
MASQUERADE_DOMAIN('mytrek.com mybeach.com')dnl
```

If you have a long list of domains or hosts, or if you want to be able to easily change those that should be masqueraded, you can place them in a file to be read by Sendmail. Specify the file with the **MASQUERADE_DOMAIN_FILE** command:

```
MASQUERADE_DOMAIN_FILE('mydomains')dnl
```

If you just want to masquerade all the hosts in your local domain, you use the **masquerade_entire_domain** feature:

```
FEATURE(masquerade_entire_domain)dnl
```

A common configuration for a local network specifies the domain name in the **MASQUERADE_AS** entry and in the **MASQUERADE_DOMAIN** entry. Using the example **myisp.com** for the domain, the entries look like this:

```
MASQUERADE_AS('mytrek.com')dnl
FEATURE(masquerade_entire_domain)dnl
```

If you want to masquerade as an ISP's mail domain, you use the ISP's domain in the **MASQUERADE_AS** entry as shown here:

```
MASQUERADE_AS('myisp.com')dnl
MASQUERADE_DOMAIN('mytrek.com')dnl
```

When mail is received from the outside bearing just the address **mytrek.com**, your network needs to know what host to send it to. This is the host designated as the mail server for the **mytrek.com** network. This information is provided by a mail exchange record (MX) in your DNS configuration that will specify that mail sent to **mytrek.com** will be handled by the mail server—in this case, **turtle.mytrek.com**:

```
mytrek.com.   IN    MX     0     turtle.mytrek.com.
```

You also have to be sure that MX relaying is enabled with the **relay_based_on_MX** feature:

```
FEATURE(relay_based_on_MX)dnl
```

All messages will appear to originate from the mail server's host. For example, if your Sendmail mail server is running on **turtle.mytrek.com**, mail sent from a localhost called **rabbit.mytrek.com** will appear to have been sent from **turtle.mytrek.com**.

You can also masquerade recipient addresses, so that mail sent to users on your localhost will be sent instead to the masqueraded address. Use the **allmasquerade** feature to enable recipient masquerading:

```
FEATURE(allmasquerade)dnl
```

Configuring Mail Servers and Mail Clients

Sendmail can be used either as a mail server, handling mail for various hosts on a network, or as a mail client, managing mail for local users on a particular host. In a simple network configuration, you have each host running Sendmail in a client configuration and one host operating as a mail server, relaying mail for the network hosts. For a local network connected to the Internet, your localhosts will run Sendmail in a client configuration, and your gateway will run Sendmail in a server configuration (though the mail server will not have to necessarily run on the gateway). The mail server relays messages from the local network hosts out to the Internet. The mail server can also be used to block unwanted access from outside hosts, such as those sending spam mail. A basic client or server Sendmail configuration involves just a few features in the **/etc/mail/sendmail.mc** file. The default configuration installed on your system allows use on a single host, managing messages between users on that host. To enable client and server use, you will need to make changes to the **/etc/mail/sendmail.mc** file.

Configuring Sendmail for a Simple Network Configuration

Some distributions, like Fedora, initially configure Sendmail to work only on the system it is running on, **localhost**. To use Sendmail to send messages to other hosts on a local network, you need to change and add settings in the **sendmail.mc** and **/etc/mail/access** files. A simple network configuration has Sendmail running on each host, handling both mail sent between users on that host and mail to and from users on other hosts. For each Sendmail server configuration, you make the changes described in the following section on simple local network configuration.

For messages sent between hosts on your network, you only need to run the Sendmail server on each, making a few changes to their Sendmail configurations. The Sendmail server on one of your hosts can be configured to handle the task of relaying messages between hosts. Using the network example described earlier, the hosts **turtle**, **rabbit**, and **lizard** will be running their own Sendmail servers. The Sendmail server on the **turtle** host will be configured to relay messages between all the hosts, itself included.

On each host on your network, edit the **/etc/mail/sendmail.mc** file and make the following changes. Comment out the **DAEMON_OPTIONS** line in the default **sendmail.mc** file by placing a **dnl** word in front of it, as shown here. Removing this feature will allow you to receive messages over your local network. This entry restricts Sendmail to the **localhost** (127.0.0.1):

```
dnl DAEMON_OPTIONS('Port=smtp,Addr=127.0.0.1, Name=MTA')dnl
```

In the **sendmail.mc** file located on the host that you want to have handle the relaying of messages, you need to also add the following line:

```
FEATURE(relay_entire_domain)dnl
```

Run the **m4** operation to install the changed configuration and then restart the server.

You can now email messages from one user to another across your network. For example, **george@turtle.mytrek.com** can now email a message to **larisa@rabbit.mytrek.com**. The local Sendmail servers will take care of sending and delivering mail both to users within their hosts and those located on other network hosts.

Configuring Sendmail for a Centralized Mail Server

Alternatively, you can set up a central mail server to handle all the mail on your network. Mail clients on various hosts can send their messages to the central mail server, which will then relay them out to the larger network or Internet. Mail could then be received at the central mail server, where clients could later retrieve it. There are several ways to set up a central mail server. One of the simplest is to run a central mail server on your gateway host and then have null client versions of the Sendmail server running on localhosts. Any mail sent from localhosts will be automatically forwarded to the central mail server. Received mail can only be delivered to the central server, usually to a POP or IMAP server also running on the central server's host. Users can then access the POP server to retrieve their mail.

For a centralized configuration, it makes sense to treat users as having their network domain as their address, rather than separate hosts in their network. Thus the user **cece** on **rabbit.mytrek.com** uses the mail address **cece@mytrek.com**, not **cece@rabbit.mytrek.com**. Users can have the same name as those on their respective hosts, but corresponding users

would be set up on the gateway host to handle received mail managed by the POP or IMAP servers.

An effective simple mail server involves several components:

- A central mail server running on the gateway host
- Each client running Sendmail as a null client
- Masquerading all mail to use the domain address only, not host addresses
- A POP or IMAP server running on the gateway host to handle received mail

Configuring a Workstation with Direct ISP Connection

If you are running a Linux system that is not part of a network but does have a direct connection to the Internet through an ISP, you can use the ISP mail servers for sending and receiving mail. Normally, you would have an SMTP mail server for outgoing mail and a POP server for incoming mail. However, you can also configure Sendmail to interface with your ISP.

Be sure to first comment out the **DAEMON_OPTIONS** option as shown in the previous sections.

Usually your ISP provides a mail server to handle mail for its hosts. To use the ISP mail server, define it with the **SMART_HOST** option. Mail will then be sent through the ISP mail server. **SMART_HOST** has the format *type*:*hostname*, where *type* is the kind of mail server used, usually SMTP. The default is relay. Define the **SMART_HOST** option to use your ISP to send and receive mail:

```
define ('SMART_HOST', 'smtp:mail.my-isp.com')dnl
```

The **SMART_HOST** option indicates a specific remote mail server that you want to have handle the relaying of your network messages. It can be an ISP mail server, as well as any mail server in a larger network.

For a dial-up connection over a modem, you can use various configuration options to control your connection. The **confMESSAGE_TIMEOUT** option lets you control how long mail can remain on the output queue, letting you keep mail until you are ready to dial in and send it. Setting the **confDELIVERY_MODE** option to **queueonly** lets you send mail only when you are ready.

The Mailer Table

The mailer table lets you route messages addressed to a specified host or domain to a particular mail server. You can use the mailer table to have mail addressed to a virtual domain routed to the mail server for your network. To reference an entire domain, prefix the domain name with a period. The host to which the mail is routed is prefixed by the mailer used, usually **smtp** for Sendmail. The following entry will route mail addressed to **.mybeach.com** to the mail server **turtle.mytrek.com**:

```
.mybeach.com        smtp:turtle.mytrek.com
```

Entries are placed in the **/etc/mail/mailertable** file. Once you have made your entries, generate the **mailertable.db** database file with the **make** command:

```
make mailertable
```

Virtual Domains: virtusertable

You can define virtual domains for your network. These virtual domains are mapped to one or more real domains by your DNS server. However, you can receive messages with mail addresses for users on your virtual domains. In this case, you need to map these addresses to users on your real domain so that the mail can be delivered to an existing location. This mapping is carried out by the virtual user table called **/etc/mail/virtusertable**. The virtual user table lets you map mail addresses for virtual domains to users on real domains. Once you have made your entries, generate the **virtusertable.db** database file with the **make** command:

```
make virtusertable
```

Security

For security, Sendmail lets you screen specific messages as well as provide authentication and encryption for Sendmail transmissions. With version 8.11, Sendmail incorporated support for the Secure Sockets Layer (SSL) and the Simple Authentication and Security Layer (SASL). Support for SSL goes by the Sendmail command **STARTTLS**, which stands for "start transport layer security." SSL provides authentication, encryption, and integrity checks for Sendmail operations. OpenSSL must first be installed to allow use of SSL encryption and authentication methods.

The SASL is implemented by the **AUTH** command and is referred to as SMTP AUTH. SASL provides authentication for mail users and servers. It can make use of already-installed Kerberos services to provide authentication.

Sendmail also provides you with the capability of screening out messages from specific domain, host, IP, and user addresses. Rules to perform such screening are kept in the **/etc/mail/access** file. You can edit this file and add your own rules. A rule consists of an address followed by an action to take. (The actions supported are listed in Table 25-4.) For example, to remove all messages from the **myannoyingad.com** domain, you enter

```
myannoyingad.com DISCARD
```

The next example rejects any message from **larisa@turtle.mycar.com** and sends a notice of the rejection:

```
larisa@turtle.mycar.com REJECT
```

Action	Description
OK	Accepts message even if other rules would reject (exception to the rules).
DISCARD	Discards the message completely.
REJECT	Rejects the message, sending a rejection notice to the sender.
RELAY	Relays messages for specified domain.
SMTP-*code message*	Code and message to be sent to sender.

TABLE 25-4 Access Actions

You can also specify an error message to return, as shown here:

```
cecelia@rabbit.mytrek.com    ERROR:"Retired yesterday"
```

To send an error message to spammers, you can include a message as shown here. The first number is an error code:

```
cyberspammer.com     ERROR:"550 We don't accept mail from spammers"
```

An **/etc/mail/access** file with the previous entries looks like the following:

```
myannoyingad.com           DISCARD
larisa@turtle.mycar.com    REJECT
cecelia@rabbit.mytrek.com  ERROR:"Retired yesterday"
cyberspammer.com           ERROR:"550 We don't accept mail from spammers"
```

Sendmail reads the access rules from a database file called **access.db**, also located in the **/etc/mail** directory. To implement your rules, you have to regenerate the **access.db** file using the access file. You can do this with the **make** command using **access** as the argument, as shown here:

```
make access
```

Sendmail then has to be restarted to read the new **access.db** file.

The access file is enabled in the **sendmail.mc** file with the **access_db** feature:

```
FEATURE('access_db')dnl
```

The access file will deny mail received from the listed addresses. However, you can also reject any mail sent to them. Additionally, you can also received mail for certain hosts on your network. You do this by enabling the **blacklist_recipients** option in the **sendmail.mc** file. This option governs recipients, whereas **access** normally governs senders. Those addresses listed will not be able to receive any mail. This feature is also used for certain administrative users that should never receive mail, such as **nobody** (the guest user) or **ftp** (the FTP user):

```
FEATURE('blacklist_recipients')dnl
```

The following example will not allow mail to be sent to **cyberspammer.com** (a recipient), nor can mail be received for **justin@lizard.mytrek.com**, **secretproject@rabbit.mytrek.com**, or **mysurfboard.com**:

```
mysurfboard.com                 ERROR:"Domain does not exist"
justin@lizard.mytrek.com        "Moved to Hawaii"
secretproject@rabbit.mytrek.com REJECT
cyberspammer.com                REJECT
```

Your distribution version of **smb.conf** may configure Sendmail to use **access_db** (as is the case with Fedora). Access is granted only to users on the localhost. If your system is being used as a mail server for a network and you have not enabled the **relay_entire_domain** feature, you will need to allow access by other hosts on your network. In the access file, you can place a **RELAY** rule for your network. The **RELAY** rule will let other hosts use your mail

server to send messages out to other hosts. This is normally done for a gateway host that needs to relay messages from a local network out to the Internet. The following example allows access from the **mytrek.com** network:

```
mytrek.com      RELAY
```

For a specific host, place an entry for it in the access file as shown here:

```
rabbit.mytrek.com    RELAY
```

To further secure Sendmail, you should disable the use of **VRFY**. This option allows remote users to try to verify the existence of a user address. This can be used to guess valid users on your system. This option is disabled with the **noverify** feature:

```
FEATURE('noverify')dnl
```

Another potential security breach is the **EXPN** option, which expands mailing lists and aliases to their actual addresses. Use the **noexpn** feature to turn it off:

```
FEATURE('noexpn')dnl
```

By default, Sendmail will refuse mail from any domain that cannot be resolved. You can override this restriction with the **accept_unresolvable_domains** feature. Sendmail will also reject mail whose addresses do not have fully qualified domain names. You can override this feature with **accept_unqualified_senders**.

POP and IMAP Server: Dovecot

The protocols Internet Mail Access Protocol (IMAP) and Post Office Protocol (POP) allow a remote server to hold mail for users who can then fetch their mail from it when they are ready. Unlike procmail, which delivers mail messages directly to a user account on a Linux system, the IMAP and POP protocols hold mail until a user accesses an account on the IMAP or POP server. The servers then transfer any received messages to the user's local mailbox. Such servers are often used by ISPs to provide Internet mail services for users. Instead of being sent directly to a user's machine, the mail resides in the IMAP or POP server until it's retrieved. Red Hat Linux and Fedora install Dovecot as both its IMAP and POP servers. Other popular IMAP and POP servers available are Qpopper, the Qmail POP server, the Washington University POP and IMAP servers, and the Courier POP and IMAP servers.

You can access the POP server from different hosts; however, when you do, all the messages are transferred to that host. They are not kept on the POP server (though you can set an option to keep them). The POP server simply forwards your messages to the requesting host. When you access your messages from a certain computer, they will be transferred to that computer and erased from the POP server. If you access your POP server again from a different computer, those previous messages will be gone.

The Internet Mail Access Protocol (IMAP) allows a remote server to hold mail for users who can then log in to access their mail. Unlike the POP servers, IMAP servers retain user mail messages. Users can even save their mail on the IMAP mail server. This has the advantage of keeping a user's mail in one centralized location accessible anywhere on

the network. Users can log in to the mail server from any host on the network and read, send, and save their mail.

Unlike POP, IMAP allows users to set up multiple folders on their mail server in which they can organize their mail. IMAP also supports the use of shared folders to which several users can access mail on a given topic.

Dovecot

Dovecot is a combination IMAP and POP server. Using its own indexing methods, Dovecot is able to handle a great deal of email traffic. It features support for SSL, along with numerous authentication methods. Password database support includes shadow passwords, LDAP, PAM, and MySQL. The **/etc/dovecot.conf** file is configured to use plain password authentication with PAM, using the **passwd** file.

Configuration options are placed in **/etc/dovecot.conf**. This file contains commented default settings with detail explanations for each. Options specific to **imap** and **pop3** are placed in their own sections. These are some basic settings to configure:

- **protocols** This can be set to **`imap`** and **`pop`**, as well as **`imaps`** and **`pops`** for SSL-encrypted connections.
- **listen** These can be set to IPv4 or IPv6 protocols; IPv6 is set by default. The listen option is set in the respective protocol sections, like **`protocol imap`** or **`protocol pop3`**.
- **auth default** section This section holds your default authentication options.
- **mechanism** in **auth** section plain by default digest-MD5 and cran-MD5 are supported, but they are not needed if you are using SSL.
- **passwd** in **auth** section**mail_location** The default mail storage method and location.

Dovecot supports either mailbox or maildir (IMAP) storage formats. The mailbox format uses single large mailbox files to hold several mail messages. Updates can be time consuming. The maildir format uses a separate file for each message, making updates much more efficient. Dovecot will automatically detect the kind of storage use, referencing the MAIL environment variable. This will be the user's **mbox** file at **/var/mail**. You can configure Dovecot to use a maildir format by setting the **`mail_location`** option to use a maildir setting, specifying the directory to use. The **`%u`** symbol can be used to represent the username, **`%h`** for the home directory. Messages will be stored in a user's **maildir** directory instead of an **mbox** file. Be sure to create the **maildir** directory and give it read, write, and execute access.

```
mail_location=maildir:/var/mail/%1u/%u/maildir
```

Other POP and IMAP Servers

Many distributions also include the Cyrus IMAP server, which you can install and use instead of Dovecot. In addition, several other IMAP and POP servers are available for use on Linux:

- The University of Washington POP and IMAP servers (**ftp.cac.washington.edu/imap**) are part of the University of Washington's **imap** RPM package. The POP server daemons are called **ipop2d** and **ipop3d**. Your Linux system then runs as a POP2 and POP3 server for your network. These servers are run through **xinetd**. The POP3 server uses the **ipop3** file in the **/etc/xinetd.d**, and the IMAP uses **imap**.

- The Cyrus IMAP server (**asg.web.cmu.edu/cyrus**) features security controls and authentication, using a private mailbox structure that is easily scalable. Designed to be run on dedicated mail servers, it is supported and maintained by Carnegie Mellon. The name of the Cyrus IMAP server daemon is **imapd**. There will be a file called **imap** in the **/etc/xinetd.d** directory.
- The Courier-IMAP server (**courier-mta.org**) is a small, fast IMAP server that provides extensive authentication support including LDAP and PAM.
- Qpopper is the Berkeley POP server (popper). Qpopper is unsupported software, currently available from Qualcomm, makers of Eudora email software. The Qpopper web page is located at the Eudora site archives (**eudora.com)**.

NOTE *The IMAP and POP servers provide SSL encryption for secure email transmissions. You can also run IMAP and POP servers using Stunnel to provide similar security. Stunnel is an SSL wrapper for daemons like* ***imapd***, ***popd***, *and even* ***pppd*** *(modem connections). In the service's* ***xinetd*** *script, you can invoke the server with the* ***stunnel*** *command instead of running the server directly.*

Spam: SpamAssassin

With SpamAssassin, you can filter sent and received email for spam. The filter examines both headers and content, drawing on rules designed to detect common spam messages. When they are detected, it then tags the message as spam, so that a mail client can then discard it. SpamAssassin will also report spam messages to spam detection databases. The version of SpamAssassin distributed for Linux is the open source version developed by the Apache project, located at **spamassassin.apache.org**. There you can find detailed documentation, FAQs, mailing lists, and even a listing of the tests that SpamAssassin performs.

SpamAssassin rule files are located at **/usr/share/spamassassin**. The files contain rules for running tests such as detecting the fake hello in the header. Configuration files for SpamAssassin are located at **/etc/mail/spamassassin**. The **local.cf** file lists systemwide SpamAssassin options such as how to rewrite headers. The **init.pre** file holds spam system configurations. The **spamassassin-spamc.rc** file will redirect all mail to the **spamc** client.

Users can set their own SpamAssassin options in their **.spamassassin/user_prefs** file. Common options include `required_scorei`, which sets a threshold for classifying a message as SPAM, numerous whitelist and blacklist options that accept and reject messages from certain users and domains, and tagging options that either rewrite or just add SPAM labels. Check the Mail::SpamAssassin::Conf Man page for details.

To configure procmail to use SpamAssassin, you need to have procmail run the **/etc/mail/spamassassin/spamassassin-spamc.rc** file. This will filter all mail through SpamAssassin. The **spamassassin-spamc.rc** file uses the **spamd** daemon, which means you have to have the SpamAssassin service running. The **spamassassin-default.rc** file runs a less efficient script to use SpamAssassin, instead of the daemon. If you want systemwide procmail filtering, you use the **/etc/procmailrc** file; whereas to implement filtering on

a per-user basis, you use a **.procmail** file in the user's home directory. Within the respective procmail files, add the following at the top:

```
INCLUDERC=/etc/mail/spamassassin/spamassassin-spamc.rc
```

Configuring Postfix for use with SpamAssassin can be complicated. A helpful tool for this task is **amavisd-new**, an interface between a mail transport agent like Sendmail or Postfix and content checkers like SpamAssassin and virus checkers. Check **ijs.si/software/amavisd** for more details.

This page intentionally left blank

26 CHAPTER

Print, News, Search, and Database Servers

Print servers have become an integrated part of every Linux system. They allow you to use any printer on your system or network. Newsgroup severs are more rare, used for setting up newsgroups for local networks or for supporting the Internet's Usenet News service. Database servers are becoming more common for managing large collections of data on local networks, as well as for Internet services.

Printer Servers: CUPS

Once treated as devices attached to a system directly, printers are now treated as network resources managed by print servers. In the case of a single printer attached directly to a system, the networking features becomes transparent and the printer appears as just one more device. On the other hand, you could easily use a print server's networking capability to let several systems use the same printer. Although printer installation is almost automatic on most Linux distributions, it helps to understand the underlying process. Printing sites and resources are listed in Table 26-1.

The Common Unix Printing System (CUPS) provides printing services. It is freely available under the GNU Public License. Though it is now included with most distributions, you can also download the most recent source-code version of CUPS from **cups.org**, which provides detailed documentation on installing and managing printers. CUPS is based on the Internet Printing Protocol (IPP), which was designed to establish a printing standard for the Internet (for more information, see **pwg.org/ipp**). Whereas the older line printer (LPD) based printing systems focused primarily on line printers, an IPP-based system provides networking, PostScript, and web support. CUPS works like an Internet server and employs a configuration setup much like that of the Apache web server. Its network support lets clients directly access printers on remote servers, without having to configure the printers themselves. Configuration needs to be maintained only on the print servers.

CUPS is the primary print server for most Linux distributions. With **libgnomecups**, GNOME now provides integrated support for CUPS, allowing GNOME-based applications to directly access CUPS printers.

Resource	Description
cups.org	Common Unix Printing System
pwg.org/ipp	Internet Printing Protocol
sourceforge.net/projects/lprng	LPRng print server

TABLE 26-1 Print Resources

Once you have installed your printers and configured your print server, you can print and manage your print queue using print clients. There are a variety of printer clients available for the CUPS server, GNOME print manager, the CUPS configuration tool, and various line printing tools like `lpq` and `lpc`. These are described in further detail later in this chapter. The CUPS configuration tool is a web-based configuration tool that can also manage printers and print jobs (open your browser and enter the URL **http://localhost:631**). A web page is displayed with entries for managing jobs, managing printers, and administrative tasks. Select the Manage Jobs entry to remove or reorder jobs you have submitted.

NOTE *Line Printer, Next Generation (LPRng) was the traditional print server for Linux and Unix systems, but it has since been dropped from many Linux distributions. You can find out more about LPRng at* ***sourceforge.net/projects/lprng****.*

Printer Devices and Configuration

Before you can use any printer, you first have to install it on a Linux system on your network. A local printer is installed directly on your own system. This involves creating an entry for the printer in a printer configuration file that defines the kind of printer it is, along with other features such as the device file and spool directory it uses. On CUPS, the printer configuration file is **/etc/cups/printers.conf**. Installing a printer is fairly simple: determine which device file to use for the printer and the configuration entries for it.

TIP *If you cannot find the drivers for your printer, you may be able to download them from the OpenPrinting database at* ***linux-foundation.org/en/OpenPrinting****. The site maintains an extensive listing of drivers.*

Printer Device Files

Linux dynamically creates the device names for printers that are installed. For parallel printers, the device names will be **lp0**, **lp1**, and **lp2**, depending on how many parallel printers are connected. The number used in these names corresponds to a parallel port on your PC; **lp0** references the LPT1 parallel port and **lp1** references the LPT2 parallel port. Serial printers will use serial ports, referenced by device files like **ttyS0**, **ttyS1**, **ttyS2**, and so on. USB-connected printers will have a Hardware Abstract Layer (HAL) device connection. HAL is designed for removable devices that can easily be attached to other connections and still be recognized.

Spool Directories

When your system prints a file, it makes use of special directories called *spool directories*. A *print job* is a file to be printed. When you send a file to a printer, a copy of it is made and placed in a spool directory set up for that printer. The location of the spool directory is obtained from the printer's entry in its configuration file. On Linux, the spool directory is located at **/var/spool/cups** under a directory with the name of the printer. For example, the spool directory for the **myepson** printer would be located at **/var/spool/cups/myepson**. The spool directory contains several files for managing print jobs. Some files use the name of the printer as their extension. For example, the **myepson** printer has the files **control.myepson**, which provides printer queue control, and **active.myepson** for the active print job, as well as **log.myepson,** which is the log file.

Installing Printers with CUPS

There are several tools available for installing CUPS printers. The easiest method is to use the GNOME CUPS tools. These are usually provided as part of the printer configuration tools included with most distributions. They can be run easily from the GNOME desktop. Alternatively you can use the CUPS web browser–based configuration tools, included with the CUPS software. Finally you can just edit the CUPS printer configuration files directly.

Configuring CUPS on GNOME

GNOME provides support for adding and configuring CUPS printers. The **gnome-cups-add** tool will detect connected printers and find the model. You can also add network printers, as well as manually configure a printer specifying a particular port. USB printers are normally detected automatically, whereas older parallel and serial printers are manually configured. To later change a printer configuration or specify printer options like paper size, right-click on its icon and select Properties. The Paper and Advanced panels let you set printing specifications, whereas the Driver and Connection panels change your printer configuration.

Configuring CUPS on KDE

KDE provides support for adding and configuring CUPS printers through the KDE Control Center. Select the Printers entry under Peripherals. The KDE Printer tool has the capability to perform many different kinds of printing such as sending faxes or saving to PDF files. USB printers that are automatically detected will be listed in the KDE Printer window. When you click on the printer entry, the Information, Jobs, Properties, and Instances panels let you manage your printer and its print jobs. The Properties panel has options for controlling user access, setting quotas, selecting a banner, and even changing your driver.

To change printer options like page size and resolution, you select the Configure entry from the Printer menu. The Printer menu also lets you disable the printer or test it. The printer toolbar also provides buttons for these commonly performed tasks. The printer manager lets you configure general features like the fonts available, the previewer to use, or the printers to display. A pop-up menu for the printer system used will have CUPS selected by default. You could switch to LPRng, if needed. Check the KDEPrint Handbook, accessible from the Documentation menu, for detailed information.

CUPS Web Browser–Based Configuration Tool

One of the easiest ways to configure and install printers with CUPS is to use the CUPS configuration tool, which is a web browser–based configuration tool.. To start the web interface, enter the following URL into your web browser:

```
http://localhost:631
```

This opens an administration screen where you can manage and add printers. You will first be asked to enter the administrator's username (usually **root**) and password (usually the root user's password).

With the CUPS configuration tool, you install a printer on CUPS through a series of web pages, each of which requests different information. To install a printer, click the Add Printer button to display a page where you enter the printer name and location. The location is the host to which the printer is connected.

Subsequent pages will prompt you to enter the model of the printer and driver, which you select from available listings. Once you have added the printer, you can configure it. Clicking the Manage Printers entry in the Administration page lists your installed printers. You can then click a printer to display a page that lets you control the printer. You can stop the printer, configure its printing, modify its installation, and even delete the printer. Clicking the Configure Printer button displays a page where you can configure how your printer prints, by specifying the resolution or paper size.

Configuration information for a printer will be stored in the **/etc/cups/printers.conf** file. You can examine this file directly, even making changes. Here is an example of a printer configuration entry. The **DeviceURI** entry specifies the device used, in this case a USB printer managed by HAL. It is currently idle, with no jobs:

```
# Printer configuration file for CUPS
# Written by cupsd
<Printer mycannon>
Info Cannon s330
Location
DeviceURI hal:///org/freedesktop/Hal/devices/usb_device_4a9_1074_300HCR_if0_printer_noserial
State Idle
StateTime 1166554036
Accepting Yes
Shared Yes
JobSheets none none
QuotaPeriod 0
PageLimit 0
KLimit 0
OpPolicy default
ErrorPolicy stop-printer
</Printer>
```

NOTE *You can perform all administrative tasks from the command line using the* **lpadmin** *command. See the CUPS documentation for more details.*

Configuring Remote Printers on CUPS

To install a remote printer that is attached to a Windows system or another Linux system running CUPS, you specify its location using special URL protocols. For another CUPS printer on a remote host, the protocol used is **ipp**, for Internet Printing Protocol, whereas for a Windows printer, it would be **smb**. Older Unix or Linux systems using LPRng would use the **lpd** protocol.

In the **cupsd.conf** file, for a remote printer, the DeviceURI entry, instead of listing the device, will have an Internet address, along with its protocol. For example, a remote printer on a CUPS server (**ipp**) would be indicated as shown here (a Windows printer would use an **smb** protocol):

```
DeviceURI ipp://mytsuff.com/printers/queue1
```

For a Windows printer, you first need to install, configure, and run Samba. (CUPS uses Samba to access Windows printers.) When you install the Windows printer on CUPS, you specify its location using the URL protocol **smb**. The user allowed to log in to the printer is entered before the hostname and separated from it by an **@** sign. On most configurations, this is the **guest** user. The location entry for a Windows printer called **myhp** attached to a Windows host named **lizard** is shown here. Its Samba share reference would be **//lizard/myhp**:

```
DeviceURI smb://guest@lizard/myhp
```

To enable CUPS on Samba, you also have to set the printing option in the **/etc/samba/smb.conf** file to **cups**, as shown here:

```
printing = cups
printcap name = cups
```

To enable CUPS to work with Samba, you have to link the **smbspool** to the CUPS **smb** spool directory:

```
ln -s /usr/bin/smbspool    /usr/cups/backend/smb
```

NOTE *To configure a shared Linux printer for access by Windows hosts, you need to configure it as an smb shared printer. You do this with Samba.*

CUPS Printer Classes

CUPS features a way to let you select a group of printers to print a job instead of selecting just one. That way, if one printer is busy or down, another printer can be automatically selected to perform the job. Such groupings of printers are called *classes.* Once you have installed your printers, you can group them into different classes. For example, you may want to group all inkjet printers into one class and laser printers into another, or you may want to group printers connected to one specific printer server in their own class. To create a class, select Classes on the Administration page and enter the name of the class. You can then add printers to it.

CUPS Configuration

CUPS configuration files are placed in the **/etc/cups** directory. These files are listed in Table 26-2. The **classes.conf**, **printers.conf**, and **client.conf** files can be managed by the web interface. The **printers.conf** file contains the configuration information for the different printers you have installed. Any of these files can be edited manually, if you wish.

cupsd.conf

The CUPS server is configured with the **cupsd.conf** file located in **/etc/cups**. You must edit configuration options manually; the server is not configured with the web interface. Your installation of CUPS installs a commented version of the **cupsd.conf** file with each option listed, though most options will be commented out. Commented lines are preceded with a **#** symbol. Each option is documented in detail. The server configuration uses an Apache web server syntax consisting of a set of directives. As with Apache, several of these directives can group other directives into blocks.

CUPS Directives

Certain directives allow you to place access controls on specific locations. These can be printers or resources, such as the administrative tool or the spool directories. Location controls are implemented with the **Location** directive. **Allow From** and **Deny From** directives can permit or deny access from specific hosts. CUPS supports both Basic and Digest forms of authentication, specified in the **AuthType** directive. Basic authentication uses a user and password. For example, to use the web interface, you are prompted to enter the root username and the root user password. Digest authentication makes use of user and password information kept in the CUPS **/etc/cups/passwd.md5** file, using MD5 versions of a username and password for authentication. The **AuthClass** directive specifies the class allowed access. The **System** class includes the root, sys, and system users. The following example shows the **Location** directive for the **/admin** resource, the administrative tool:

```
<Location /admin>

AuthType Basic
AuthClass System

## Restrict access to local domain
Order Deny,Allow
Deny From All
Allow From 127.0.0.1

</Location>
```

Filename	Description
classes.conf	Contains configurations for different local printer classes
client.conf	Lists specific options for specified clients
cupsd.conf	Configures the CUPS server, **cupsd**
printers.conf	Contains printer configurations for available local printers

Table 26-2 CUPS Configuration Files

CUPS Command Line Print Clients

Once a print job is placed on a print queue, you can use any of several print clients to manage the printing jobs on your printer or printers, such as Klpq, the GNOME Print Manager, and the CUPS Printer Configuration tool for CUPS. You can also use several command line print CUPS clients. These include the **lpr**, **lpc**, **lpq**, and **lprm** commands. The Printer System Switcher moves you from one set to the other. With these clients, you can print documents, list a print queue, reorder it, and remove print jobs, effectively canceling them. For network connections, CUPS features an encryption option for its commands, **-E**, to encrypt print jobs and print information sent from a network. Table 26-3 shows various printer management methods.

NOTE *The command line clients have the same name, and much the same syntax, as the older LPR and LPRng command line clients used in Unix and older Linux systems.*

lpr

The **lpr** client submits a job, and **lpd** then takes it in turn and places it on the appropriate print queue; **lpr** takes as its argument the name of a file. If no printer is specified, then the default printer is used. The **-P** option enables you to specify a particular printer. In the next example, the user first prints the file **preface** and then prints the file **report** to the printer with the name **myepson**:

```
$ lpr preface
$ lpr -P myepson report
```

Printer Management	Description
GNOME Print Manager	GNOME print queue management tool (CUPS).
CUPS Configuration Tool	Prints, manages, and configures CUPS.
lpr *options file-list*	Prints a file, copies the file to the printer's spool directory, and places it on the print queue to be printed in turn. **-P** *printer* prints the file on the specified printer.
lpq *options*	Displays the print jobs in the print queue. **-P** *printer* prints the queue for the specified printer. **-l** prints a detailed listing.
lpstat *options*	Displays printer status.
lprm *options printjob-id* **or** *printer*	Removes a print job from the print queue. You identify a particular print job by its number as listed by **lpq**. **-P** *printer* removes all print jobs for the specified printer.
lpc	Manages your printers. At the **lpc>** prompt, you can enter commands to check the status of your printers and take other actions.

TABLE 26-3 CUPS Print Clients

lpc

You can use **lpc** to enable or disable printers, reorder their print queues, and re-execute configuration files. To use **lpc**, enter the command **lpc** at the shell prompt. You are then given an **lpc>** prompt at which you can enter **lpc** commands to manage your printers and reorder their jobs. The **status** command with the name of the printer displays whether the printer is ready, how many print jobs it has, and so on. The **stop** and **start** commands can stop a printer and start it back up. The printers shown depend on the printers configured for a particular print servers. A printer configured on CUPS will only show if you have switched to CUPS.

```
# lpc
lpc> status myepson
myepson:
 printer is on device 'hal' speed -1
 queuing is enabled
 printing is enabled
 1 entry in spool area
```

lpq and lpstat

You can manage the print queue using the **lpq** and **lprm** commands. The **lpq** command lists the printing jobs currently on the print queue. With the **-P** option and the printer name, you can list the jobs for a particular printer. If you specify a username, you can list the print jobs for that user. With the **-l** option, **lpq** displays detailed information about each job. If you want information on a specific job, simply use that job's ID number with **lpq**. To check the status of a printer, use **lpstat**.

```
# lpq
myepson is ready and printing
Rank    Owner  Jobs  File(s)          Total Size
active  chris    1   report           1024
```

lprm

The **lprm** command enables you to remove a print job from the queue, erasing the job before it can be printed. The **lprm** command takes many of the same options as **lpq**. To remove a specific job, use **lprm** with the job number. To remove all printing jobs for a particular printer, use the **-P** option with the printer name. **lprm** with no options removes the job printing currently. The following command removes the first print job in the queue (use **lpq** to obtain the job number):

```
# lprm 1
```

CUPS Command Line Administrative Tools

CUPS provides command line administrative tools like **lpadmin**, **lpoptions**, **lpinfo**, **enable**, **disable**, **accept**, and **reject**. The **enable** and **disable** commands start and stop print queues directly, whereas the **accept** and **reject** commands start and stop particular jobs. The **lpinfo** command provides information about printers, and **lpoptions** lets you set printing options. The **lpadmin** command lets you perform administrative tasks like adding printers and changing configurations. CUPS administrative tools are listed in Table 26-4.

Administrative Tool	Description
lpadmin	CUPS printer configuration
lpoptions	Sets printing options
enable	Activates a printer
disable	Stops a printer
accept	Allows a printer to accept new jobs
reject	Prevents a printer from accepting print jobs
lpinfo	Lists CUPS devices available

TABLE 26-4 CUPS Administrative Tools

lpadmin

You can use the **lpadmin** command to either set the default printer or configure various options for a printer. You can use the **-d** option to specify a particular printer as the default destination. Here **myepson** is made the default printer:

```
lpadmin -d myepson
```

The **-p** option lets you designate a printer for which to set various options. The following example sets printer description information:

```
lpadmin -p myepson  -D  Epson550
```

Certain options let you control per-user quotas for print jobs. The **job-k-limit** option sets the size of a job allowed per user, **job-page-limit** sets the page limit for a job, and **job-quota-period** limits the number of jobs with a specified time frame. The following command sets a page limit of 100 for each user:

```
lpadmin -p myepson  -o job-page-limit=100
```

User access control is determined with the **-u** option with an **allow** or **deny** list. Users allowed access are listed following the **allow**: entry, and those denied access are listed with a **deny:** entry. Here access is granted to **chris** but denied to **aleina** and **larisa**.

```
lpadmin -p myepson -u allow:chris  deny:aleina,larisa
```

Use **all** or **none** to permit or deny access to all or no users. You can create exceptions by using **all** or **none** in combination with user-specific access. The following example allows access to all users except **justin**:

```
lpadmin -p myepson  -u allow:all   deny:justin
```

lpoptions

The **lpoptions** command lets you set printing options and defaults that mostly govern how your print jobs will be printed. For example, you can set the color or page format to be used with a particular printer. Default settings for all users are maintained by the root user

in the **/etc/cups/lpoptions** file, and each user can create his or her own configurations, which are saved in his or her **.lpoptions** files. The **-l** option lists current options for a printer, and the **-p** option designates a printer (you can also set the default printer to use with the **-d** option).

```
lpoptions -p myepson -l
```

Printer options are set using the **-o** option along with the option name and value, **-o** *option*=*value*. You can remove a printer option with the **-r** option. For example, to print on both sides of your sheets, you can set the **sides** option to **two-sided**:

```
lpoptions -p myepson -o sides=two-sided
```

To remove the option, use **-r**.

```
lpoptions -p myepson -r sides
```

To display a listing of available options, check the standard printing options in the CUPS Software Manual at **cups.org**.

enable and disable

The **enable** command starts a printer, and the **disable** command stops it. With the **-c** option, you can cancel all jobs on the printer's queue, and the **-r** option broadcasts a message explaining the shutdown.

```
disable myepson
```

accept and reject

The **accept** and **reject** commands let you control access to the printer queues for specific printers. The **reject** command prevents a printer from accepting jobs, whereas **accept** allows new print jobs.

```
reject myepson
```

lpinfo

The **lpinfo** command is a handy tool for letting you know what CUPS devices and drivers are available on your system. Use the **-v** option for devices and the **-m** option for drivers.

```
lpinfo -m
```

News Servers

News servers provide Internet users with Usenet news services. They have their own TCP/IP protocol, the Network News Transfer Protocol (NNTP). On most Linux systems, including Red Hat, the InterNetNews (INN) news server provides news services (**isc.org**). In addition, servers exist that provide better access to Internet resources.

News Servers: INN

The InterNetNews (INN) news server accesses Usenet newsfeeds, providing news clients on your network with the full range of newsgroups and their articles. Newsgroup articles are transferred using NNTP, and servers that support this protocol are known as *NNTP servers*. INN was written by Rich Salz and is currently maintained and supported by the Internet Software Consortium (ISC). You can download current versions from its website at **isc.org**. INN is also included with most Linux distributions. The documentation directory for INN in **/usr/share/doc** contains extensive samples. The primary program for INN is the **innd** daemon.

INN Configuration Files

Various INN configuration files can be found in **/etc/news**, including **inn.conf**, **storage.conf**, **readers.conf**, and **incoming.conf**; **inn.conf** sets options for INN, and **incoming.conf** holds the hosts from which you receive newsfeeds. Place entries for remote hosts in the **readers .conf** file to allow them access to your news server. Actual newsfeeds are managed in directories in the **/var/spool/news** directory. Here you will find directories such as **article**, which holds newsgroup articles, **outgoing** for articles being posted by your users to newsgroups, and **overview**, which holds summary information about articles. Correct configuration of INN

File	Description
inn.conf	General INN configuration file.
incoming.conf	Specifies hosts from which newsfeeds are received.
cycbuff.conf	Configures buffers used in cnfs storage format.
storage.conf	Defines storage classes. These consist of a storage method and the newsgroups that use it. Storage methods are the storage formats: tradspool, timehash, timecaf, and cnfs. An additional method, trash, throws out the articles.
expire.ctl	Sets the expiration policy for articles on the news server.
readers.conf	Designates hosts whose users can access the news server with newsreaders.
ovdb.conf	Configures ovdb storage method for overviews.
newsfeeds	Defines how your news server feeds articles to other news servers.
moderated	Moderated newsgroups.
active	Supported newsgroups.
history	Record of posted articles.
innfeed.conf	Configures newsfeed processes for innfeed.
innreport.conf	Configures the innreport utility for generating log-based reports.
buffindexed.conf	Configures overview buffer for buffindexed method.

Table 26-5 INN Configuration Files

can be a complex and time-consuming process, so be sure to consult references and online resources, such as the documents. When you change configurations, be sure to restart the INN server. An **innd** script is in the **/etc/rc.d/init.d** directory, which has similar arguments to the web **httpd** script. You can use **start**, **restart**, and **stop** arguments with the **innd** script to start, restart, and stop the INN server.

TIP *There is a Man page for each configuration file in INN, providing detailed information on how to configure their features.*

inn.conf

On many distributions, a basic **inn.conf** file is already set up for you with default settings. Several of the initial parameters you will have to set yourself, such as **domain**, which holds the domain name for your server; **pathhost**, in which you specify the name for your newsreader as you want it to appear in the Path header field for news articles you post; and **server**, in which you specify your newsreader's IP or fully qualified domain name address, as in **mynews.mytrek.com**. Different Path options have already been set up for you defining the location of different INN directories, such as **patharticles**, set to **/var/spool/news** articles that holds your newsgroup articles, and **pathetc**, set to **/etc/news** for your configuration files.

Storage Formats

Storage formats for the vast number of news articles that are often downloaded and accessed are a central concern for a full-scale news server like INN. INN lets you choose among four possible storage formats: tradspool, timehash, timecaf, and cnfs. The tradspool format is the traditional method whereby articles are arranged in a simple directory structure according to their newsgroups. This is known to be very time consuming to access and store. timehash stores articles in directories organized by the time they were received, making it easier to remove outdated articles. timecaf is similar to timehash, but articles received at a given time are placed in the same file, making access much faster. cnfs stores articles into buffer files that have already been set up. When a buffer file becomes full, the older articles are overwritten by new ones as they come in. This is an extremely fast method, since no new files are created. There is no need to set maximum article limits, but there is also no control on how long an article is retained. In the **storage.conf** file, storage formats are assigned as storage methods to different newsgroups.

Newsreader Access

Users access your news server using newsreaders. You can place controls on users with options in the **readers.conf** file. Control is specified in two components: authentication and access definitions. The authentication definition creates a user category and the hosts and their authentication tools for users. The access definition applies restrictions to a user category, such as what newsgroups can be accessed and whether posting articles is allowed.

Overviews

INN also supports overviews. These are summaries of articles that readers can check, instead of having to download the entire article to see what it is. Overviews have their own storage

methods: tradindexed, buffindexed, and ovdb. You specify the one you want to use in the ovmethod feature in **inn.conf**. tradindexed is fast for readers but difficult for the server to generate. buffindexed is fast for news servers but slow for readers. ovdb uses Berkeley DB database files and is very fast for both but uses more disk space. If you choose ovdb, you can set configuration parameters for it in **ovdb.conf**.

INN Implementation

On many distributions, a **news** user is already created with a newsgroup for use by your INN daemon and sets up the news directories in **/var/spool/news**. INN software also installs **cron** scripts, which are used to update your news server, removing old articles and fetching new ones. These are usually placed in the **/etc/cron.daily** directory, though they may reside anywhere. `inn-cron-expire` removes old articles, and `inn-cron-rnews` retrieves new ones. `inn-cron-nntpsend` sends articles posted from your system to other news servers.

INN also includes several support programs to provide maintenance and crash recovery and to perform statistical analysis on server performance and usage. cleanfeed implements spam protection, and innreport generates INN reports based on logs. INN also features a very strong filter system for screening unwanted articles.

NOTE *Leafnode is an NNTP news server designed for small networks that may have slow connections to the Internet. You can obtain the Leafnode software package along with documentation from its website at* ***leafnode.org****. Along with the Leafnode NNTP server, the software package includes several utilities such as Fetchnews, Texpire, and Newsq that send, delete, and display news articles. slrnpull is a simple single-user version of Leafnode that can be used only with the slrn newsreader.*

Database Servers: MySQL and PostgreSQL

Two fully functional database servers are included with most Linux distributions, MySQL and PostgreSQL. MySQL is by far the more popular of the two, though PostgreSQL is noted for providing more features. Recently, the MySQL AB project added MaxDB, formerly SAP DB, which provides capabilities comparable to many professional-level database management systems. This chapter will cover how to set up and manage a MySQL database and will offer a brief introduction to PostgreSQL. You can learn more about these products through the sites listed in Table 26-6.

Database	Resource
MySQL	**mysql.com**
PostgreSQL	**postgresql.org**
MaxDB	**mysql.com**

TABLE 26-6 Database Resources

Relational Database Structure

MySQL and PostgreSQL both use a relational database structure. Essentially, this means data is placed in tables, with identifier fields used to relate the data to entries in other tables. Each row in the table is a record, each with a unique identifier, like a record number. The connections between records in different tables are implemented by special tables that associate the unique identifiers from records in one table with those of another. Relational database theory and implementation are subjects beyond the scope of this chapter.

A simple, single-table database has no need for a unique identifier. A simple address book listing names and addresses is an example of a single-table database. However, most databases access complex information of different types, related in various ways. Instead of having large records with repeated information, you divide the data in different tables, each holding the unique instance of the data. This way, data is not repeated; you have only one table that holds a single record for a person's name, rather than repeating that person's name each time the data references him or her. The relational organization then takes on the task of relating one piece of data to another. This way, you can store a great deal of information using relatively small database files.

Though there are many ways to implement a relational database, a simple rule of thumb is to organize data into tables where you have a unique instance of each item of data. Each record is given a unique identifier, usually a number. To associate the records in one table with another, you create tables that associate their identifiers.

SQL

The SQL query language is the language used by most relational database management systems (RDBMSs), including both MySQL and PostgreSQL. Though many RDBMSs use administrative tools to manage databases, on Linux MySQL and PostgreSQL, you still have to use the SQL commands directly. Common SQL commands that you may use are listed in Table 26-7. The commands are often written in uppercase by convention, though they can be in lowercase.

Command	Description
CREATE DATABASE *name*	Creates a database.
CREATE TABLE *name* (*fields*, ..)	Creates a table within a database, specifying fields.
INSERT INTO *table-name* VALUES (*value list*)	Creates and inserts a record into a table.
INSERT INTO *table-name* VALUES (*value list*), (*value list*), ...	Inserts multiple records at once.
SELECT *field* FROM *table-name* WHERE *value*	Search operation, selects certain records in a table based on a value in a specified field.
USE *database*	Uses a particular database; following commands will operate on it.

Table 26-7 SQL Commands

Using the previously described relational database, the following command will create the database:

```
CREATE DATABASE myphotos
```

Before performing any operations on a database, you first access it with the USE command.

```
USE myphotos
```

The tables are created using the CREATE TABLE command; the fields for each table are listed within parentheses following the table name. For each field, you need to specify a name, data type, and other options, such as whether it can have a null value.

```
CREATE TABLE names (
    personid INT(5) UNSIGNED NOT NULL,
    name VARCHAR(20) NOT NULL,
    street VARCHAR(30) NOT NULL,
    phone CHAR(8)
     );
```

To insert a record into a table, you can use the INSERT INTO command, though many databases support using data files that can be read all at once. To add records, you use the INSERT INTO command with the table name followed by the VALUES option, which is followed in turn by a comma-delimited list of values, one for each field. Character values are quoted with single quotes. The list is enclosed in parentheses. If you have not done so previously, you access the database with the USE command.

```
INSERT INTO names VALUES (1, 'justin','111 mordor','555-7543');
```

Once values are added to the tables, you can search them with the SELECT command, specifying field, table name, and the value to be searched.

```
SELECT phone FROM names WHERE phone='555-7543';
```

MySQL

MySQL is structured on a client/server model with a server daemon (**mysqld**) filling requests from client programs. MySQL is designed for speed, reliability, and ease of use. It is meant to be a fast database management system for large databases and, at the same time, a reliable one, suitable for intensive use.

To create databases, you use the standard SQL language. User access can be controlled by assigning privileges.

MySQL Configuration

The MySQL supports three different configuration files, one for global settings, another for server-specific settings, and an optional one for user-customized settings.

- The **/etc/my.cnf** configuration file is used for global settings applied to both clients and servers. The **/etc/my.cnf** file provides information such as the data directory (**/var/lib/mysql**) and log file (**/var/log/mysql.log**) locations, as well as the server base directory (**/var/lib**).

- The **/var/lib/mysql/my.cnf** file is used for server settings only.
- The **.my.cnf** file allows users to customize their access to MySQL. It is located in a user's home directory. Note that this is a dot file.

Sample configuration **my.cnf** files can be found in the **mysql-server** directory in **/usr/share/doc**. The **mysql-server** directory lists configurations for small, medium, large, and huge implementations. The administrative manual is located in the **mysql** directory for **/usr/share/doc**. It is in the info format. Use **info mysql** to start it and the arrow and ENTER keys to move through the menus. Here you can find more information about different options.

Global Configuration:/etc/my.cnf

MySQL specifies options according to different groups, usually the names of server tools. The options are arranged in group segments. The group name is placed within brackets, and options applied to it follow. The default **/etc/my.cnf** file is shown here:

```
[mysqld]
datadir=/var/lib/mysql
socket=/var/lib/mysql/mysql.sock

[mysql.server]
user=mysql
basedir=/var/lib

[safe_mysqld]
err-log=/var/log/mysqld.log
pid-file=/var/run/mysqld/mysqld.pid
```

MySQL global options are listed in the **/etc/my.cnf** file. Options are set up according to groups that control different behaviors of the MySQL server: **mysqld** for the daemon, **mysql.server** for server options, and **safe_mysqld** for the MySQL startup script. The **datadir** directory, **/var/lib/mysql**, is where your database files will be placed. Server tools and daemons are located in the **basedir** directory, **/var/lib**, and the user that MySQL will run as has the name **mysql**, as specified in the **user** option.

A client group will set up options to be sent to clients, such as the port and socket to use to access the MySQL database.

```
[client]
port=3306
socket=/var/lib/mysql/mysql.sock
```

To see what options are currently set for both client and server, you run **mysqld** directly with the **--help** option.

```
/usr/libexec/mysqld --help
```

User Configuration: .my.cnf

Users who access the database server will have their own configuration file in their home directory: **.my.cnf**. Here the user can specify connection options such as the password used to access the database and the connection timeouts.

```
[client]
password=mypassword

[mysql]
no-auto-rehash
set-variable = connect_timeout=2

[mysql-hotcopy]
interactive-timeout
```

MySQL Tools

MySQL provides a variety of tools (as shown in Table 26-8), including server, client, and administrative tools. Backups can be handled with the **mysldump** command. The **mysqlshow** command will display a database, just as issuing the SQL command SELECT *.* does, and **mysqlimport** can import text files, just like LOAD INFILE.

MySQL Management with mysql and mysqladmin

To manage your MySQL database, you use **mysql** as the **root** user. The **mysql** client starts up the MySQL monitor. As the root user, you can enter administrative commands to create databases and database tables, add or remove entries, as well as carry out standard client tasks such as displaying data.

Log in as the root user and open a terminal window. Then enter the **mysql** command. This will start a MySQL monitor shell with a **mysql>** prompt. Be sure to end your commands with a semicolon; otherwise, the monitor will provide an indented arrow prompt waiting for added arguments. In the monitor, the semicolon, not the ENTER key, ends commands.

```
 # mysql -u root -p
mysql>
```

If you have set up a MySQL root user, you can use the **-u** root with the **-p** option. You will be prompted for a password.

```
# mysql -u root -p
```

Once the **mysql** client has started, you can use the **status** command to check the status of your server and **show databases** to list current databases.

```
mysql> status;
mysql> show databases;
```

Command	Description
mysqld	MySQL server
mysql	MySQL client
mysqladmin	Creates and administers databases
mysqldump	Database backup
mysqlimport	Imports text files
mysqlshow	Displays databases

TABLE 26-8 MySQL Commands

Initially two databases set up by MySQL for its own management are displayed: mysql and test. The mysql database holds MySQL user information, and the test database is used to test the server.

PostgreSQL

PostgreSQL is based on the POSTGRES database management system, though it uses SQL as its query language. POSTGRES is a next-generation research prototype developed at the University of California, Berkeley. You can find more information on it from the PostgreSQL website at **postgresql.org**. PostgreSQL is an open source project, developed under the GPL.

PostgreSQL is often used to provide database support for Internet servers with heavy demands, such as web servers. With a few simple commands, you can create relational database tables. Use the `createuser` command to create a PostgreSQL user that you can then log in to the server with. You can then create a database with the `createdb` command and construct relational tables using the `create table` directive. With an `insert` command, you can add records and then view them with the `select` command. Access to the server by remote users is controlled by entries in the **pg_hba.conf** file located in PostgreSQL directory, usually **/var/lib/pgsql**.

The Red Hat Linux edition of PostgreSQL also includes the Red Hat Database Graphical tools to easily manage and access PostgreSQL databases. With the administrator tool, you can browse and manage databases; the Visual Explain tool analyzes query processes; and the Control Center lets you manage databases on servers.

NOTE *The search and indexing server ht://Dig enables document searches of websites and FTP sites (**htdig.org**). With it, you can index documents and carry out complex search requests.*

PART VII

System Administration

CHAPTER 27
Basic System Administration

CHAPTER 28
Managing Users

CHAPTER 29
File Systems

CHAPTER 30
RAID and LVM

CHAPTER 31
Devices and Modules

CHAPTER 32
Kernel Administration

CHAPTER 33
Backup Management

This page intentionally left blank

27
CHAPTER

Basic System Administration

Linux is designed to serve many users at the same time, providing an interface between the users and the system with its resources, services, and devices. Users have their own shells through which they interact with the operating system, but you may need to configure the operating system itself in different ways. You may need to add new users, devices like printers and scanners, and even file systems. Such operations come under the heading of system administration. The person who performs such actions is referred to as either a *system administrator* or a *superuser.* In this sense, there are two types of interaction with Linux: regular users' interactions, and those of the superuser, who performs system administration tasks. The chapters in this part of the book cover operations such as changing system runlevels, managing users, configuring printers, adding file systems, and compiling the kernel. You perform most of these tasks only rarely, such as adding a new printer or mounting a file system. Other tasks, such as adding or removing users, you perform on a regular basis. Basic system administration covers topics such as system access by superusers, selecting the runlevel to start, system configuration files, and performance monitoring.

With Linux, you have the ability to load different versions of the Linux kernel as well as other operating systems that you have installed on your system. The task of selecting and starting up an operating system or kernel is managed by a boot management utility, the Grand Unified Bootloader (GRUB). This is a versatile tool, letting you load operating systems that share the same disk drive, as well as letting you choose from different Linux kernels that may be installed on the same Linux system.

Superuser Control: The Root User

To perform system administration operations, you must first have access rights, such as the correct password, that enable you to log in as the root user, making you the superuser. Because a superuser has the power to change almost anything on the system, such a password is usually a carefully guarded secret, changed very frequently, and given only to those whose job it is to manage the system. With the correct password, you can log in to the system as a system administrator and configure the system in different ways. You can start up and shut down the system, as well as change to a different operating mode, such as a single-user mode. You can also add or remove users, add or remove whole file systems, back up and restore files, and even designate the system's name and address.

NOTE *If SELinux is enabled, superuser access will be controlled by SELinux rules.*

To become a superuser, you log in to the *root user account.* This is a special account reserved for system management operations with unrestricted access to all components of your Linux operating system. You can log in as the root user from either the GUI (graphical user interface) login screen or the command line login prompt. You then have access to all administrative tools. Using a GUI interface like GNOME, the root user has access to a number of distribution GUI administrative tools. If you log in from the command line interface, you can run corresponding administrative commands like **rpm** to install packages or **useradd** to add a new user. From your GUI desktop, you can also run command line administrative tools using a terminal window. The command line interface for the root user uses a special prompt, the sharp sign, **#**. In the next example, the user logs in to the system as the root user and receives the **#** prompt.

```
login: root
password:
#
```

Root User Password

As the root user, you can use the **passwd** command to change the password for the root login, as well as for any other user on the system. The **passwd** command will check your password with Pluggable Authentication Modules (PAM), to see if you've selected one that can be easily cracked.

```
# passwd root
New password:
Re-enter new password:
#
```

You must take precautions to protect your root password. Anyone who gains access as the root user will have complete control over your system. The online manual for the **passwd** command provides detailed recommendations for handling and choosing your password. For example, never store your password in a file on your system, and never choose one based on any accessible information, such as your phone number or date of birth. A basic guideline is to make your password as complex as possible, using a phrase of several words with numbers and upper- and lowercase letters, yet something you can still remember easily so that you never have to write it down. You can access the **passwd** online manual page with the command

```
# man passwd
```

Root User Access: su

While you are logged in to a regular user account, it may be necessary for you to log in as the root and become a superuser. Ordinarily, you would have to log out of your user account first, and then log in to the root. Instead, you can use the **su** command (switch user) to log in directly to the root while remaining logged in to your user account. If you are using a GUI desktop like GNOME, you can enter the **su** command from a terminal window, or use ALT-CTRL-F1 to switch to a command line interface (ALT-CTRL-F10 returns you to the GUI interface).

Command	Description
su root	Logs a superuser into the root from a user login; the superuser returns to the original login with a CTRL-D.
sudo *command*	Restricts administrative access to specified users.
passwd login-name	Sets a new password for the login name.
telinit *runlevel*	Changes the system runlevels.
shutdown *options time*	Shuts down the system.
date	Sets the date and time for the system.

TABLE 27-1 Basic System Administration Tools

A CTRL-D or **exit** command returns you to your own user login. When you are logged in as the root, you can use **su** to log in as any user, without providing the password. In the next example, the user is logged in already. The **su** command then logs in as the root user, making the user a superuser. Some basic superuser commands are shown in Table 27-1.

```
$ pwd
/home/chris
$su
 password:
# cd
# pwd
/root
# exit
$
```

CAUTION *For security reasons, Linux distributions do not allow the use of* **su** *in a Telnet session to access the root user. For SSH- or Kerberos-enabled systems, secure login access is provided using slogin (SSH) and rlogin (Kerberos version).*

Controlled Administrative Access: sudo

With the sudo tool you can allow ordinary users to have limited root user–level administrative access for certain tasks. This allows other users to perform specific superuser operations without having full root level control. You can find more about sudo at **sudo.ws**. To use sudo to run an administrative command, the user precedes the command with the **sudo** command. The user is issued a time-sensitive ticket to allow access.

```
sudo date
```

The first time you issue a **sudo** command during a login session, you will be prompted to enter your administrative password.

Access is controlled by the **/etc/sudoers** file. This file lists users and the commands they can run, along with the password for access. If the **NOPASSWD** option is set, then users will not need a password. **ALL**, depending on the context, can refer to all hosts on your network, all root-level commands, or all users.

NOTE *Some distributions like Ubuntu deny direct **root** access by default, and only allow administrative **root** access through **sudo** commands.*

To make changes or add entries, you have to edit the file with the special sudo editing command **visudo**. This invokes the Vi editor to edit the **/etc/sudoers** file. Unlike a standard editor, **visudo** will lock the **/etc/sodoers** file and check the syntax of your entries. You are not allowed to save changes unless the syntax is correct. If you want to use a different editor, you can assign it to the EDITOR shell variable.

A **sudoers** entry has the following syntax:

```
user   host=command
```

The host is a host on your network. You can specify all hosts with the **ALL** term. The command can be a list of commands, some or all qualified by options such as whether a password is required. To specify all commands, you can also use the ALL term. The following gives the user **george** full root-level access to all commands on all hosts:

```
george  ALL = ALL
```

In addition, you can let a user run as another user on a given host. Such alternate users are placed within parentheses before the commands. For example, if you want to give **george** access to the **beach** host as the user **mydns**, you use the following:

```
george beach = (mydns) ALL
```

By default sudo will deny access to all users, including the root. For this reason, the default **/etc/sudoers** file sets full access for the root user to all commands. The **ALL=(ALL) ALL** entry allows access by the root to all hosts as all users to all commands.

```
root   ALL=(ALL)   ALL
```

To specify a group name, you prefix the group with a % sign, as in %**mygroup**. This way, you can give the same access to a group of users. The **/etc/sudoers** file contains samples for a %**wheel** group.

To give **robert** access on all hosts to the **date** command, you use

```
robert ALL=/usr/bin/date
```

If a user wants to see what commands he or she can run, that user uses the **sudo** command with the **-l** option.

```
sudo -l
```

System Time and Date

You probably set the time and date when you first installed your system and should not need to do so again. If you entered the time incorrectly or moved to a different time zone, though, you can set the system time and date using either the shell **date** command or desktop tools such as the KDE Date & Time tool or the GNOME Time and Date Settings (time-admin).

In addition, many Linux distributions provide their own date and time utilities. The desktop and distribution tools provide an easy to use GUI interface.

You can use the **date** command on your root user command line to set the date and time for the system. As an argument to **date**, you list (with no delimiters) the month, day, time, and year. In the next example, the date is set to 2:59 P.M., April 6, 2008 (04 for April, 06 for the day, 1459 for the time, and 03 for the year 2008):

```
# date 0406145908
Sun Mar 6 02:59:27 PST 2008
```

Most date and time tools also provide an option to use time servers to set the time automatically. The Network Time Protocol (NTP) allows a remote server to set the date and time instead of using local settings. NTP allows for the most accurate synchronization of your system's clock. It is often used to manage the time and date for networked systems, freeing the administrator from having to synchronize clocks manually. You can download current documentation and NTP software from the **ntp.org** site.

A date and time tool will let you choose to enable NTP and select the server to use. NTP servers operate through pools that will randomly select an available server to increase efficiency. A set of pools designated may already be installed for you by your distribution. If access with one pool is slow, you change to another. The **pool.ntp.org** pool servers support worldwide access. Pools for specific geographical locations can be found at the NTP Public Services Project site (Time Servers link), **ntp.isc.org**. A closer server could be faster.

Scheduling Tasks: cron

Scheduling regular maintenance tasks, such as backups, is managed by the **cron** service on Linux, implemented by a **cron** daemon. A daemon is a continually running server that constantly checks for certain actions to take. These tasks are listed in the **crontab** file. The **cron** daemon constantly checks the user's **crontab** file to see if it is time to take these actions. Any user can set up a **crontab** file of his or her own. The root user can set up a **crontab** file to take system administrative actions, such as backing up files at a certain time each week or month.

The easiest way to schedule tasks is to use the desktop **cron** tool. For KDE you can use the KCRON tool and for GNOME you can use GNOME Schedule. Both provide panels for choosing the month, date, and time for a process, though you will have to manually enter the command you want run, as if on a command line. A listing of **cron** entries lets you modify or delete tasks. If you have an open ended operation, be sure to also schedule a command to shut it down.

The name of the **cron** daemon is **crond**. Normally it is started automatically when your system starts up. You can set this feature using a management service like **chkconfig** (Fedora and SUSE), **services-admin** (GNOME), or or **sysv-rc-conf**. You can also start and stop the **crond** service manually, which you may want to do for emergency maintenance or during upgrades.

crontab Entries

A **crontab** entry has six fields: the first five are used to specify the time for an action, while the last field is the action itself. The first field specifies minutes (0–59), the second field specifies the hour (0–23), the third field specifies the day of the month (1–31), the fourth

field specifies the month of the year (1–12, or month prefixes like *Jan* and *Sep*), and the fifth field specifies the day of the week (0–6, or day prefixes like *Wed* and *Fri*), starting with 0 as Sunday. In each of the time fields, you can specify a range, specify a set of values, or use the asterisk to indicate all values. For example, **1-5** for the day-of-week field specifies Monday through Friday. In the hour field, **8, 12, 17** would specify 8 A.M., 12 noon, and 5 P.M. An ***** in the month-of-year field indicates every month. The format of a **crontab** field follows:

```
minute  hour  day-month  month  day(s)-week  task
```

The following example backs up the **projects** directory at 2:00 A.M. every weekday:

```
0 2 * * 1-5   tar cf /home/backp /home/projects
```

The same entry is listed here again using prefixes for the month and weekday:

```
0 2 * * Mon-Fri tar cf /home/backp /home/projects
```

To specify particular months, days, weeks, or hours, you can list them individually, separated by commas. For example, to perform the previous task on Sunday, Wednesday, and Friday, you can use **0,3,5** in the day-of-week field, or their prefix equivalents, **Sun,Wed,Fri**.

```
0 2 * * 0,3,5   tar cf /home/backp /home/projects
```

cron also supports comments. A comment is any line beginning with a **#** sign.

```
# Weekly backup for Chris's projects
0 2 * * Mon-Fri  tar cf /home/backp /home/projects
```

Environment Variables for cron

The **cron** service also lets you define environment variables for use with tasks performed. Linux defines variables for **SHELL**, **PATH**, **HOME**, and **MAILTO**. **SHELL** designates the shell to use tasks, in this case the BASH shell. **PATH** lists the directories where programs and scripts can be found. This example lists the standard directories, **/usr/bin** and **/bin**, as well as the system directories reserved for system applications, **/usr/sbin** and **/sbin**. **MAILTO** designates to whom the results of a task are to be mailed. By default, these are mailed to the user who schedules it, but you can have the results sent to a specific user, such as the administrator's email address, or an account on another system in a network. **HOME** is the home directory for a task, in this case the top directory.

```
SHELL=/bin/bash
PATH=/sbin:/bin:/usr/sbin:/usr/bin
MAILTO=root
HOME=/
```

The cron.d Directory

On a heavily used system, the **/etc/crontab** file can become crowded easily. There may also be instances where certain entries require different variables. For example, you may need to run some task under a different shell. To help better organize your **crontab** tasks, you can place **crontab** entries in files within the **cron.d** directory. The files in the **cron.d** directory all

contain **crontab** entries of the same format as **/etc/crontab**. They may be given any name. They are treated as added **crontab** files, with **cron** checking them for tasks to run. For example, Linux installs a **sysstat** file in the **cron.d** that contains **crontab** entries to run tools to gather system statistics.

The crontab Command

You use the **crontab** command to install your entries into a **crontab** file. To do this, first create a text file and type your **crontab** entries. Save this file with any name you want, such as **mycronfile**. Then, to install these entries, enter **crontab** and the name of the text file. The **crontab** command takes the contents of the text file and creates a **crontab** file in the **/var/spool/cron** directory, adding the name of the user who issued the command. In the following example, the root user installs the contents of **mycronfile** as the root's **crontab** file. This creates a file called **/var/spool/cron/root**. If a user named **justin** installs a **crontab** file, it creates a file called **/var/spool/cron/justin**. You can control use of the **crontab** command by regular users with the **/etc/cron.allow** file. Only users with their names in this file can create **crontab** files of their own. Conversely, the **/etc/cron.deny** file lists those users denied use of the **cron** tool, preventing them from scheduling tasks. If neither file exists, access is denied to all users. If a user is not in an **/etc/cron.allow** file, access is denied. However, if the **/etc/cron.allow** file does not exist, and the **/etc/cron.deny** file does, then all users not listed in **/etc/cron.deny** are automatically allowed access.

```
# crontab mycronfile
```

Editing in cron

Never try to edit your **crontab** file directly. Instead, use the **crontab** command with the **-e** option. This opens your **crontab** file in the **/var/spool/cron** directory with the standard text editor, such as Vi (**crontab** uses the default editor as specified by the **EDITOR** shell environment variable). To use a different editor for **crontab**, change the default editor by assigning the editor's program name to the **EDITOR** variable and exporting that variable. Normally, the **EDITOR** variable is set in the **/etc/profile** script. Running **crontab** with the **-l** option displays the contents of your **crontab** file, and the **-r** option deletes the entire file. Invoking **crontab** with another text file of **crontab** entries overwrites your current **crontab** file, replacing it with the contents of the text file.

Organizing Scheduled Tasks

You can organize administrative **cron** tasks into two general groups: common administrative tasks that can be run at regular intervals, or specialized tasks that need to be run at a unique time. Unique tasks can be run as entries in the **/etc/crontab** file, as described in the next section. Common administrative tasks, though they can be run from the **/etc/crontab** file, are better organized into specialized **cron** directories. Within such directories, each task is placed in its own shell script that will invoke the task when run. For example, there may be several administrative tasks that all need to be run each week on the same day, say if maintenance for a system is scheduled on a Sunday morning. For these kinds of tasks, **cron** provides several specialized directories for automatic daily, weekly, monthly, and yearly tasks. Each contains a **cron** prefix and a suffix for the time interval. The **/etc/cron.daily** directory is used for tasks that need to be performed every day, whereas weekly tasks can be placed in the **/etc/cron.weekly** directory. The **cron** directories are listed in Table 27-2.

cron Command and Tools	Description
`crontab` *options filename*	With *filename* as an argument, installs `crontab` entries in the file to a **crontab** file; these entries are operations executed at specified times `-e` Edits the **crontab** file `-l` Lists the contents of the **crontab** file `-r` Deletes the **crontab** file
Kcron	KDE GUI interface **cron** management tool
Schedule	GNOME GUI interface **cron** management tool
cron Files and Directories	
/etc/crontab	System `crontab` file, accessible only by the root user
/etc/cron.d	Directory containing multiple `crontab` files, accessible only by the root user
/etc/cron.hourly	Directory for tasks performed hourly
/etc/cron.daily	Directory for tasks performed daily
/etc/cron.weekly	Directory for tasks performed weekly
/etc/cron.monthly	Directory for tasks performed monthly
/etc/cron.yearly	Directory for tasks performed yearly
/etc/cron.allow	Users allowed to submit **cron** tasks
/etc/cron.deny	Users denied access to **cron**

TABLE 27-2 cron Command, Tools, Files and Directories

Running cron Directory Scripts

Each directory contains scripts that are all run at the same time. The scheduling for each group is determined by an entry in the **/etc/crontab** file. The actual execution of the scripts is performed by the **/usr/bin/run-parts** script, which runs all the scripts and programs in a given directory. Scheduling for all the tasks in a given directory is handled by an entry in the **/etc/crontab** file. Linux provides entries with designated times, which you may change for your own needs. The default **crontab** file is shown here, with times for running scripts in the different **cron** directories. Here you can see that most scripts are run at about 4 A.M. either daily (4:02), Sunday (4:22), or first day of each month (4:42). Hourly scripts are run one minute after the hour.

```
SHELL=/bin/bash
PATH=/sbin:/bin:/usr/sbin:/usr/bin
MAILTO=root
HOME=/
# run-parts
01 * * * * root run-parts /etc/cron.hourly
02 4 * * * root run-parts /etc/cron.daily
22 4 * * 0 root run-parts /etc/cron.weekly
42 4 1 * * root run-parts /etc/cron.monthly
```

TIP *Scripts within a **cron** directory are run alphabetically. If you need a certain script to run before any others, you may have to alter its name. One method is to prefix the name with a numeral. For example, in the **/cron.weekly** directory, the **anacron** script is named **0anacron** so that it will run before any others.*

Keep in mind though that these are simply directories that contain executable files. The actual scheduling is performed by the entries in the **/etc/crontab** file. For example, if the weekly field in the `cron.weekly` `crontab` entry is changed to * instead of `0`, and the monthly field to `1` (`22 4 1 * *` instead of `22 4 * * 0`), tasks in the **cron.weekly** file will end up running monthly instead of weekly.

cron Directory Names

The names used for these directories are merely conventions. They have no special meaning to the **cron** daemon. You could, in fact, create your own directory, place scripts within it, and schedule run-parts to run those scripts at a given time. In the next example, scripts placed in the **/etc/cron.mydocs** directory will run at 12 noon every Wednesday.

```
* 12 * * 3 root run-parts /etc/cron.mydocs
```

Anacron

For a system that may normally be shut down during times that **cron** is likely to run, you may want to supplement **cron** with **anacron**, which activates only when scheduled tasks need to be executed. For example, if a system is shut down on a weekend when **cron** jobs are scheduled, then the jobs will not be performed; **anacron**, however, checks to see what jobs need to be performed when the system is turned on again, and then runs them. It is designed only for jobs that run daily or weekly.

For **anacron** jobs, you place `crontab` entries in the **/etc/anacrontab** file. For each scheduled task, you specify the number of intervening days when it is executed (7 is weekly, 30 is monthly), the time of day it is run (numbered in minutes), a description of the task, and the command to be executed. For backups, the command used is the **tar** operation.

System Runlevels: telinit, initab, and shutdown

A Linux system can run in different levels, depending on the capabilities you want to give it. For example, you can run your system at an administrative level, locking out user access. Normal full operations are activated by simply running your system at a certain level of operational capability such as supporting multiuser access or graphical interfaces. These levels (also known as states or modes) are referred to as *runlevels,* the level of support that you are running your system at.

Runlevels

A Linux system has several runlevels, numbered from 0 to 6. When you power up your system, you enter the default runlevel. Runlevels 0, 1, and 6 are special runlevels that perform specific functions. Runlevel 0 is the power-down state and is invoked by the `halt` command to shut down the system. Runlevel 6 is the reboot state—it shuts down the system and reboots. Runlevel 1 is the single-user state, which allows access only to the

superuser and does not run any network services. This enables you, as the administrator, to perform administrative actions without interference from others.

Other runlevels reflect how you want the system to be used. These may differ depending on the distribution you have. The runlevels described here are used on Red Hat, Fedora, SUSE, and similar distributions. Debian and Ubuntu have slightly different runlevel configuration, using runlevel 2 for graphical logins and the remainder as user defined.

On Red Hat/Fedora and similar distributions, runlevel 2 is a partial multiuser state, allowing access by multiple users, but without network services like NFS or **xinetd** (extended Internet services daemon). This level is useful for a system that is not part of a network. Both runlevel 3 and runlevel 5 run a fully operational Linux system, with multiuser support and remote file sharing access. They differ in terms of the interface they use. Runlevel 3 starts up your system with the command line interface (also known as the text mode interface). Runlevel 5 starts up your system with an X session, running the X Window System server and invoking a graphical login, using display managers such as gdm or xdm. If you choose to use graphical logins during installation, runlevel 5 will be your default runlevel. Linux provides two keyboard sequences to let you switch between the two during a login session: CTRL-ALT-F1 changes from the graphical interface (runlevel 5) to the command line interface (runlevel 3), and CTRL-ALT-F7 changes from the command line interface to the graphical interface. The runlevels are listed in Table 27-3.

Changing runlevels can be helpful if you have problems at a particular runlevel. For example, if your video card is not installed properly, then any attempt to start up in runlevel 5 will likely fail, as this level immediately starts your graphical interface. Instead, you should use the command line interface, runlevel 3, to fix your video card installation.

System Runlevels (states)	Description
0	Halt (do *not* set the default to this level); shuts down the system completely.
1	Administrative single-user mode; denies other users access to the system but allows root access to the entire multiuser file system. Startup scripts are not run. (Use **s** or **S** to enter single-user mode with startup scripts run).
2	Multiuser, without network services like NFS, **xinetd**, and NIS (the same as 3 but without networking).
3	Full multiuser mode with login to command line interface; allows remote file sharing with other systems on your network. Also referred to as the *text mode state.*
4	Unused.
5	Full multiuser mode that starts up in an X session, initiating a graphical login; allows remote file sharing with other systems on your network (same as 3, but with graphical login). On Ubuntu, this is runlevel 2.
6	Reboots; shuts down and restarts the system (do *not* set the default to this).

TABLE 27-3 System Runlevels (States), Red Hat, Fedora, SUSE, and Similar Distributions

TIP *You can use the single-user runlevel (1) as a recovery mode state, allowing you to start up your system without running startup scripts for services like DNS. This is helpful if your system hangs when you try to start such services. Networking is disabled, as well as any multiuser access. You can also use* **`linux -s`** *at the boot prompt to enter runlevel 1. If you want to enter the single-user state and also run the startup scripts, you can use the special* **`s`** *or* **`S`** *runlevel.*

Runlevels in initab

When your system starts up, it uses the default runlevel as specified in the default **`init`** entry in the **/etc/inittab** file. For example, if your default init runlevel is 5 (the graphical login), the default **`init`** entry in the **/etc/inittab** file would be

```
id:5:initdefault:
```

You can change the default runlevel by editing the **/etc/inittab** file and changing the **`init`** default entry. Editing the **/etc/inittab** file can be dangerous. You should do this with great care. As an example, if the default runlevel is 3 (command line), the entry for your default runlevel in the **/etc/inittab** file should look like the following:

```
id:3:initdefault:
```

You can change the 3 to a 5 to change your default runlevel from the command line interface (3) to the graphical login (5). Change only this number and nothing else.

```
id:5:initdefault:
```

TIP *If your* ***/etc/inittab*** *file becomes corrupted, you can reboot and enter* **`linux single`** *at the boot prompt to start up your system, bypassing the* ***inittab*** *file. You can then edit the file to fix it.*

Changing Runlevels with telinit

No matter what runlevel you start in, you can change from one runlevel to another with the **`telinit`** command. If your default runlevel is 3, you power up in runlevel 3, but you can change to, say, runlevel 5 with **`telinit 5`**. The command **`telinit 0`** shuts down your system. In the next example, the **`telinit`** command changes to runlevel 1, the administrative state:

```
# telinit 1
```

On Red Hat, Fedora, SUSE, and similar distributions, one common use of telinit to change runlevels is when you need to install a software package that requires that the X server be shut down. This is the case with the graphics drivers obtained directly from Nvidia or ATI. You first have to change to runlevel 3 with a **`telinit 3`** command, shutting down the X server, and then you can install the graphics driver.

```
telinit 3
```

After installation, you can return to the X server and its GUI interface with the **`telinit 5`** command.

```
telinit 5
```

Keep in mind that the distribution packages for the Nvidia and ATI graphics packages are preferred to using those obtained directly from Nvidia or ATI.

NOTE *On Debian, Ubuntu, and similar distributions, the desktop version invokes the X server at all primary runlevels. Using* ***telinit*** *to switch to another runlevel will not let you shut down the X server. This requires shutting down the display managers (your login screen), by running either the* ***gdm*** *(GNOME) or* ***kdm*** *(KDE) service scripts with the* ***stop*** *option. Use the command* ***sudo /etc/init.d/gdm stop*** *to shut down the X server.*

The **telinit** command is really a symbolic link (another name for a command) to the **init** command. The **init** command performs the actual startup operations and is automatically invoked when your system starts up. Though you could use **init** to change runlevels, it is best to use **telinit**. When invoked as **telinit**, **init** merely changes runlevels.

To use init, enter the **init** command and the runlevel number on a command line. If you are in runlevel 3 (command line), the following places you in runlevel 5 (graphical interface):

```
init 5
```

The runlevel Command

Use the **runlevel** command to see what state you are currently running in. It lists the previous state followed by the current one. If you have not changed states, the previous state will be listed as N, indicating no previous state. This is the case for the state you boot up in. In the next example, the system is running in state 3, with no previous state change:

```
# runlevel
N 3
```

Shutdown

Although you can power down the system with the **telinit** command and the 0 state, you can also use the **shutdown** command. The **shutdown** command has a time argument that gives users on the system a warning before you power down. You can specify an exact time to shut down or a period of minutes from the current time. The exact time is specified by *hh***:***mm* for the hour and minutes. The period of time is indicated by a **+** and the number of minutes. The **shutdown** command takes several options with which you can specify how you want your system shut down. The **-h** option, which stands for halt, simply shuts down the system, whereas the **-r** option shuts down the system and then reboots it. In the next example, the system is shut down after ten minutes:

```
# shutdown -h +10
```

To shut down the system immediately, you can use **+0** or the word **now**. The following example shuts down the system immediately and then reboots:

```
# shutdown -r now
```

With the **shutdown** command, you can include a warning message to be sent to all users currently logged in, giving them time to finish what they are doing before you shut them down.

```
# shutdown -h +5 "System needs a rest"
```

If you do not specify either the **-h** or the **-r** option, the **shutdown** command shuts down the multiuser mode and shifts you to an administrative single-user mode. In effect, your system state changes from 3 (multiuser state) to 1 (administrative single-user state). Only the root user is active, allowing the root user to perform any necessary system administrative operations with which other users might interfere.

TIP *You can also shut down your system from the GNOME or KDE desktops.*

The shutdown options are listed in Table 27-4.

Command	Description
shutdown [-rkhncft] *time* [*warning-message*]	Shuts the system down after the specified time period, issuing warnings to users; you can specify a warning message of your own after the time argument; if neither **-h** nor **-r** is specified to shut down the system, the system sets to the administrative mode, runlevel state 1.
Argument	
Time	Has two possible formats: an absolute time in the format *hh*:*mm*, with *hh* as the hour (one or two digits) and *mm* as the minute (in two digits), or the format +*m*, with *m* as the number of minutes to wait; the word **now** is an alias for **+0**.
Option	
-t sec	Tells **init** to wait *sec* seconds between sending the warning and the kill signals before changing to another runlevel.
-k	Doesn't actually shut down; only sends the warning messages to everybody.
-r	Reboots after shutdown, runlevel state 6.
-h	Halts after shutdown, runlevel state 0.
-n	Doesn't call **init** to do the shutdown; you do it yourself.
-f	Skips file system checking (fsck) on reboot.
-c	Cancels an already running shutdown; no time argument.

TABLE 27-4 System Shutdown Options

NOTE *You can select certain services to run and the runlevel at which to run them. Most services are servers like a web server or proxy server. Other services provide security, such as SSH or Kerberos. Most distributiona have their own tools for managing servers. Two of the most common are* **`chkconfig`** *(Fedora and SUSE) and sysv-rc-conf. Some tools like* ***services-admin*** *and* ***rcconf*** *turn on and off services for default runlevels.*

System Directories

Your Linux file system is organized into directories whose files are used for different system functions (see Table 27-5). For basic system administration, you should be familiar with the system program directories where applications are kept, the system configuration directory (**/etc**) where most configuration files are placed, and the system log directory (**/var/log**) that holds the system logs, recording activity on your system. Other system directories are covered in their respective chapters, with many discussed in Chapter 29.

Directory	Description
/bin	System-related programs
/sbin	System programs for specialized tasks
/lib	System and application libraries
/etc	Configuration files for system and network services and applications
/home	User home directories and server data directories, such as web and FTP site files
/mnt	Where CD-ROM and floppy disk file systems are mounted (Chapter 29)
/var	System directories whose files continually change, such as logs, printer spool files, and lock files (Chapter 29)
/usr	User-related programs and files; includes several key subdirectories, such as **/usr/bin**, **/usr/X11**, and **/usr/doc**
/usr/bin	Programs for users
/dev	Dynamically generated directory for device files (Chapter 31)
/etc/X11	X Window System configuration files
/usr/share	Shared files
/usr/share/doc	Documentation for applications
/tmp	Directory for system temporary files
/var/log	Logging directory
/var/log/	System logs generated by **syslogd**
/var/log/audit	Audit logs generated by **auditd**

TABLE 27-5 System Directories

Program Directories

Directories with "bin" in the name are used to hold programs. The **/bin** directory holds basic user programs, such as login shells (BASH, TCSH, and ZSH) and file commands (`cp`, `mv`, `rm`, `ln`, and so on). The **/sbin** directory holds specialized system programs for such tasks as file system management (`fsck`, `fdisk`, `mkfs`) and system operations like shutdown and startup (`init`). The **/usr/bin** directory holds program files designed for user tasks. The **/usr/sbin** directory holds user-related system operations, such as `useradd` to add new users. The **/lib** directory holds all the libraries your system uses, including the main Linux library, `libc`, and subdirectories such as **modules**, which holds all the current kernel modules.

Configuration Directories and Files

When you configure different elements of your system, such as users, applications, servers, or network connections, you use configuration files kept in certain system directories. Configuration files are placed in the **/etc** directory.

Configuration Files: /etc

The **/etc** directory holds your system, network, server, and application configuration files. Here you can find the **fstab** file listing your file systems, the **hosts** file with IP addresses for hosts on your system, and **/etc/profile**, the system wide default BASH shell configuration file. This directory includes various subdirectories, such as **/etc/apache** for the Apache web server configuration files, **/etc/X11** for the X Window System and window manager configuration files, and **/etc/udev** for rules to generate device files in **/dev**. You can configure many applications and services by directly editing their configuration files, though it is best to use a corresponding administration tool. Table 27-6 lists several commonly used configuration files found in the **/etc** directory.

System Logs: /var/log and syslogd

Various system logs for tasks performed on your system are stored in the **/var/log** directory. Here you can find logs for mail, news, and all other system operations, such as web server logs. The **/var/log/messages** file is a log of all system tasks not covered by other logs. This usually includes startup tasks, such as loading drivers and mounting file systems. If a driver for a card failed to load at startup, you will find an error message for it here. Logins are also recorded in this file, showing you who attempted to log in to what account. The **/var/log/maillog** file logs mail message transmissions and news transfers.

NOTE *To view logs, you can use the GNOME System Log Viewer.*

syslogd and syslog.conf

The **syslogd** daemon manages all the logs on your system and coordinates with any of the logging operations of other systems on your network. Configuration information for **syslogd** is held in the **/etc/syslog.conf** file, which contains the names and locations for your system log files. Here you find entries for **/var/log/messages** and **/var/log/maillog**, among others. Whenever you make changes to the **syslog.conf** file, you need to restart the **syslogd** daemon.

File	Description
/etc/bashrc	Default shell configuration file Bash shell
/etc/group	A list of groups with configurations for each
/etc/fstab	Automatically mounts file systems when you start your system
/boot/grub/menu.lst	The GRUB configuration file for the GRUB boot loader (linked to by **/etc/grub.conf** on Red Hat)
/etc/inittab	Sets the default state, as well as terminal connections
/etc/profile	Default shell configuration file for users
/etc/modprobe.conf	Modules on your system to be automatically loaded
/etc/motd	System administrator's message of the day
/etc/mtab	Currently mounted file systems
/etc/passwd	User password and login configurations
/etc/services	Services run on the system and the ports they use
/etc/shadow	User-encrypted passwords
/etc/shells	Shells installed on the system that users can use
/etc/sudoers	sudo configuration to control administrative access
/etc/termcap	A list of terminal type specifications for terminals that could be connected to the system
/etc/xinetd.conf	Xinetd server configuration
Directory	
/etc/cron	**cron** scripts
/etc/cups	CUPS printer configuration files
/etc/init.d	Service scripts for distribution that support SysV Init scripts.
/etc/mail	Sendmail configuration files
/etc/openldap	Configuration for Open LDAP server
/etc/rc.d	Startup scripts for different runlevels
/etc/skel	Versions of initialization files, such as **.bash_profile**, which are copied to new users' home directories
/etc/X11	X Window System configuration files
/etc/xinetd.d	Configuration scripts for services managed by Xinetd server
/etc/udev	Rules for generating devices (Chapter 31)
/etc/hal	Rules for generating removable devices (Chapter 31)

Table 27-6 Common System Configuration Files and Directories

Entries in syslog.conf

An entry in **syslog.conf** consists of two fields: a *selector* and an *action*. The selector is the kind of service to be logged, such as mail or news, and the action is the location where messages are to be placed. The action is usually a log file, but it can also be a remote host or a pipe to another program. The kind of service is referred to as a *facility*. The **syslogd** daemon has several terms it uses to specify certain kinds of service (see Table 27-7). A facility can be further

Facilities	Description
`authpriv`	Security/authorization messages (private)
`cron`	Clock daemon (**cron** and at) messages
`daemon`	Other system daemon messages
`kern`	Kernel messages
`lpr`	Line printer subsystem messages
`mail`	Mail subsystem messages
`mark`	Internal use only
`news`	Usenet news subsystem messages
`syslog`	Syslog internal messages
`user`	Generic user-level messages
`uucp`	UUCP subsystem messages
`local0` through `local7`	Reserved for local use
Priorities	**Description**
`debug`	7, debugging messages, lowest priority
`info`	6, informational messages
`notice`	5, notifications, normal, but significant condition
`warning`	4, warnings
`err`	3, error messages
`crit`	2, critical conditions
`alert`	1, alerts, action must be taken immediately
`emerg`	0, emergency messages, system is unusable, highest priority
Operators	**Description**
`*`	Match all facilities or priorities in a sector
`=`	Restrict to a specified priority
`!`	Exclude specified priority and higher ones
`/`	A file to save messages to
`@`	A host to send messages to
`|`	A FIFO pipe to send messages to

Table 27-7 Syslogd Facilities, Priorities, and Operators

qualified by a priority. A *priority* specifies the kind of message generated by the facility; **syslogd** uses several designated terms to indicate different priorities. A *sector* is constructed from both the facility and the priority, separated by a period. For example, to save error messages generated by mail systems, you use a sector consisting of the **mail** facility and the **err** priority, as shown here:

```
mail.err
```

To save these messages to the **/var/log/maillog** file, you specify that file as the action, giving you the following entry:

```
mail.err /var/log/maillog
```

The **syslogd** daemon also supports the use of * as a matching character to match either all the facilities or all the priorities in a sector: **cron.*** matches on all **cron** messages no matter what the priority; ***.err** matches on error messages from all the facilities; and ***.*** matches on all messages. The following example saves all mail messages to the **/var/log/maillog** file and all critical messages to the **/var/log/mycritical** file:

```
mail.* /var/log/maillog
*.crit /var/log/mycritical
```

Priorities

When you specify a priority for a facility, all messages with a higher priority are also included. Thus the **err** priority also includes the **crit**, **alert**, and **emerg** priorities. If you just want to select the message for a specific priority, you qualify the priority with the **=** operator. For example, **mail.=err** will select only error messages, not **crit**, **alert**, or **emerg** messages. You can also restrict priorities with the **!** operator. This will eliminate messages with the specified priority and higher. For example, **mail.!crit** will exclude **crit** messages, as well as the higher **alert** and **emerg** messages. To specifically exclude all the messages for an entire facility, you use the **none** priority; for instance, **mail.none** excludes all mail messages. This is usually used when you're defining several sectors in the same entry.

You can list several priorities or facilities in a given sector by separating them with commas. You can also have several sectors in the same entry by separating them with semicolons. The first example saves to the **/var/log/messages** file all messages with **info** priority, excluding all mail and authentication messages (**authpriv**). The second saves all **crit** messages and higher for the **uucp** and **news** facilities to the **/var/log/spooler** file:

```
*.info;mail.none;news.none;authpriv.none /var/log/messages
uucp,news.crit /var/log/spooler
```

Actions and Users

In the action field, you can specify files, remote systems, users, or pipes. An action entry for a file must always begin with a / and specify its full pathname, such as **/var/log/messages**. To log messages to a remote host, you simply specify the hostname preceded by an **@** sign. The following example saves all kernel messages on **rabbit.trek.com**:

```
kern.* @rabbit.trek.com
```

To send messages to users, you list their login names. The following example will send critical news messages to the consoles for the users **chris** and **aleina**:

```
news.=crit chris,aleina
```

You can also output messages to a named pipe (FIFO). The pipe entry for the action field begins with a |. The following example pipes kernel debug messages to the named pipe |**/usr/adm/debug**:

```
kern.=debug |/usr/adm/debug
```

An Example for /etc/syslog.conf

The default **/etc/syslog.conf** file is shown here. Messages are logged to various files in the **/var/log** directory.

/etc/syslog.conf

```
# Log all kernel messages to the console.
#kern.*                                 /dev/console
# Log anything (except mail) of level info or higher.
# Don't log private authentication messages!
*.info;mail.none;news.none;authpriv.none;cron.none     /var/log/messages

# The authpriv file has restricted access.
authpriv.*                          /var/log/secure
# Log all the mail messages in one place.
mail.*                               /var/log/maillog
# Log cron stuff.
cron.*                               /var/log/cron
# Everybody gets emergency messages
*.emerg                             *
# Save mail and news errors of level err and higher in a special file.
uucp,news.crit                 /var/log/spooler
# Save boot messages also to boot.log
local7.*                             /var/log/boot.log
# INN
news.=crit                          /var/log/news/news.crit
news.=err                          /var/log/news/news.err
news.notice                       /var/log/news/news.notice
```

The Linux Auditing System: auditd

The Linux Auditing System provides system call auditing. The auditing is performed by a server called **auditd**, with logs saved to the **/var/log/audit** directory. It is designed to complement SELinux, which saves its messages to the **auditd** log in the **/var/log/audit/audit.log** file. The audit logging service provides specialize logging for services like SELinux. Logs are located at **/var/log/audit**. To refine the auditing, you can create audit rules to check certain system calls like those generated by a specific user or group.

Configuration for **auditd** is located in both the **/etc/auditd.conf** and the **/etd/sysconfig/auditd** files. Primary configuration is handled with **/etc/auditd.conf**, which holds such options as the log file name, the log format, the maximum size of log files, and actions to

take when disk space diminishes. See the **auditd.conf** Man page for a detailed description of all options. The **/etc/sysconfig/auditd** file sets server startup options and locale locations such as **en_US**.

The audit package includes the **auditd** server and three commands: **autrace**, **ausearch**, and **auditctl**. You use **ausearch** to query the audit logs. You can search by various IDs; by process, user, group, or event, as well as by filename or even time or date. Check the **ausearch** Man page for a complete listing. **autrace** is a specialized tool that lets you trace a specific process. It operates similar to **strace**, recording the system calls and actions of a particular process.

You can control the behavior of the **auditd** server with the **auditctl** tool. With **auditctl** you can turn auditing on or off, check the status, and add audit rules for specific events. Check the **auditctl** Man page for a detailed description.

Audit rules are organized into predetermined lists with a specific set of actions for system calls. Currently there are three lists: task, entry, and exit, and three actions: never, always, and possible. When adding a rule, the list and action are paired, separated by a comma, as in:

```
exit,always
```

To add a rule you use the **-a** option. With the **-S** option you can specify a particular system call, and with the **-F** option you can specify a field. There are several possible fields you can use, such as loginuid (user login ID), pid (process ID), and exit (system call exit value). For a field you specify a value, such as **loginuid=510** for the user with a user login ID of 510. The following rule, as described in the documentation, checks all files opened by a particular user:

```
auditctl -a exit,always  -S open -F loginuid=510
```

Place rules you want loaded automatically in the **/etc/auditd.rules**. The **sample.rules** file in the **/usr/share/doc/auditd*** directory lists rule examples. You can also create a specific file of audit rules and use **auditctl** with the **-R** option to read the rules from it.

Performance Analysis Tools and Processes

Linux treats each task performed on your system as a process, which is assigned a number and a name. You can examine these processes and even stop them. Linux provides several tools for examining processes as well as your system performance. Easy monitoring is provided by the GNOME System Monitor. Other tools are also available, such as GKrellM and KSysguard.

A number of utilities on your system provide detailed information on your processes, as well as other system information such as CPU and disk use (see Table 27-8). Although these tools were designed to be used on a shell command line, displaying output in text lines, several now have KDE and GNOME versions that provide a GUI interface for displaying results and managing processes.

Performance Tool	Description
vmstat	Performance of system components
top	Listing of most CPU-intensive processes
free	Listing of free RAM memory
sar	System activity information
iostat	Disk usage
GNOME System Monitor	System monitor for processes and usage monitoring
GKrellM	Stackable, flexible, and extensible monitoring tool that displays information on a wide variety of system, network, and storage operations, as well as services, easily configurable with themes
KDE Task Manager and Performance Monitor	KDE system monitor for processes and usage monitoring
Frysk	Monitoring tool for system processes
System Tap	Tool to analyze performance bottlenecks
GNOME Power Manager	Manages power efficiency features of your system
cpuspeed	Implements CPU speed reduction during idle times (AMD Cool and Quiet).

TABLE 27-8 Performance Tools

GNOME System Monitor

The GNOME System Monitor displays system information and monitoring system processes. There are four panels, System information, Processes, Resources, and File Systems (see Figure 27-1). The Resources panel displays graphs for CPU, Memory and Swap memory, and Network usage. Your File Systems panel lists your file systems, where they are mounted, and their type, as well as the amount of disk space used and how much is free. The Processes panel lists your processes, letting you sort or search for processes. You can use field buttons to sort by name, process ID, user, and memory. The View pop-up menu lets you select all processes, just your own, or active processes. You can easily stop any process by selecting it and then clicking the End Process button. Right-clicking an item displays actions you can take on the process such as stopping or hiding it. The Memory Maps display, selected from the View menu, shows information on virtual memory, inodes, and flags.

The ps Command

From the command line, you can use the **ps** command to list processes. With the **-aux** option, you can list all processes. Piping the output to a **grep** command with a pattern enables you to search for a particular process. A pipe funnels the output of a preceding command as input to a following command. The following command lists all X Window System processes:

```
ps -aux | grep 'X'
```

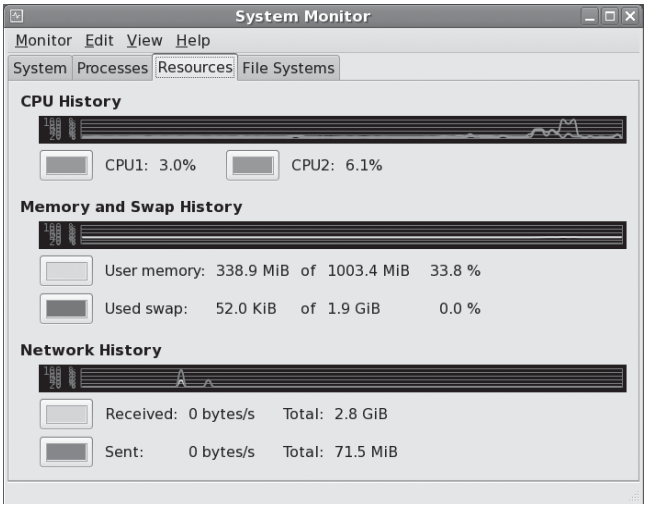


FIGURE 27-1 GNOME System Monitor

vmstat, top, free, Xload, iostat, and sar

The **vmstat** command outputs a detailed listing indicating the performance of different system components, including CPU, memory, I/O, and swap operations. A report is issued as a line with fields for the different components. If you provide a time period as an argument, it repeats at the specified interval—usually a few seconds. The **top** command provides a listing of the processes on your system that are the most CPU intensive, showing what processes are using most of your resources. The listing is in real time and updated every few seconds. Commands are provided for changing a process's status, such as its priority.

The **free** command lists the amount of free RAM memory on your system, showing how much is used and how much is free, as well as what is used for buffers and swap memory. **Xload** is an X Window System tool showing the load, CPU, and memory; **iostat** displays your disk usage; and **sar** shows system activity information.

System Tap

System Tap is a diagnostic tool for providing information about complex system implementations. It essentially analyzes performance bottlenecks, letting you home in on where a problem could be located. System Tap relies on Kprobes (Kernel Dynamic Probes), which allows kernel modules to set up simple probes.

Frysk

Frysk is a specialized complex monitoring tool for system processes that allows you to set up specific monitoring tasks, focusing on particular applications and selecting from a set of observer processes to provide information about exit notification, system calls, and execution. You can also create your own customized observers for processes. Find out more about Frysk at **sourceware.redhat.com/frysk**.

GNOME Power Manager

The GNOME Power Manager is designed to take full advantage of the efficiency features available on both laptops and desktops. It supports tasks like reducing CPU frequency, dimming the display, shutting down unused hard drives, and automatic shutdown or suspension. See **gnome.org/projects/gnome-power-manager/index.html** for a detailed description. The GNOME Power Manager is integrated with HAL (Hardware Abstraction Layer) and Dbus to detect hardware states and issue hardware notifications. Hardware notifications are issued using notification icons for devices, such as the battery icon. The notification icons are located on the panel. A tooltip on the battery icon will show how much time you have left.

Power management preferences for laptops lets you set sleep, brightness, and action settings for battery and direct power (AC). For desktops you can set the inactivity time for putting the display to sleep or suspending the system. You can access the preferences from the System | Preferences | More Preferences | Power Management item.

Features like Cool and Quit for Athlon CPUs and Pentium M frequency controls are handled separately by the cpuspeed service. CPU frequency reporting tools are provided by cpufreq and gkrellm-freq packages. The cpufreq package installs two applications, cpu-info and cpu-set, which will require CPU frequency drivers set in the kernel configuration and can be compiled as modules.

GKrellM

GKrellM is a GTK-based set of small stackable monitors for various system, network, and device operations. A title bar at the top of the stack will display the host name of your system. By default, GKrellM displays the host name, system time, CPU load, process chart, disk access, network devices like **eth0**, memory use, and a mail check. You can change the chart display of a monitor, its height, for instance, by right-clicking it to show a display options panel.

Each monitor will have a title bar, showing, for instance, CPU for CPU load, Disk for disk access, and Mem for memory. To configure the monitor, right-click its title bar. This will display the configuration panels for that task. For example, the Disk configuration will let you choose particular hard disks and partitions to monitor. The full configuration window will be displayed, showing a sidebar with configuration menus, with the built-in menu expanded to the selected monitor.

See the **gkrellm** Man page for a detailed description of all monitor configuration options. The GKrellM site, **gkrellm.net**, offers resources for documentation, program support, and themes. GKrellM is installed with the built-in plug-ins and with the WiFi plug-in. For added plug-ins and themes, you can download and install additional packages. These will provide an extensive set of plug-ins and themes for your use, including radio controls; a keyboard LED monitor; and a large list of themes such as marble, Gotham City, and shiny metal blue. There are several plug-in packages: media, misc, and utils. Plug-ins can be downloaded directly from **gkrellm.net** and themes from **muhri.net**, as well as from distribution affiliated software repositories. User themes can be placed in a user's **.gkrellm2/themes** directory.

GKrellM Configuration

You can open the configuration window directly by clicking any monitor and pressing F1. Alternatively, you can open the main menu and select the Configuration entry. To open the main menu, either press F2 or right-click the top monitor. You can use this same menu to

move through themes or to quit GKrellM. The configuration window shows a sidebar of configuration entries such as General, Builtins, Plugins, and Themes. Panels to the right let you set the configuration options. The General panel shows global options such as displaying the host name, the overall window size, and window priority.

The Builtins entry will expand to show a list of all the monitors that GKrellM can display. This list is extensive, including monitors as varied as fan and heat sensors, clock, CPU load, Internet connections, mail notification, and battery use.

The Plugins entry will expand to the list of installed plug-ins. Click the Plugins entry directly to see a list of available plug-ins with check boxes. To install a plug-in, click its check box. As you install other plug-in packages, more will be listed. When installed, a plug-in will appear in the expanded plug-in menu. You can select a plug-in there to display its configuration panels.

Plug-ins for system-wide use are located in the **/usr/lib/gkrellm2/plugins** directory, and themes are located at **/usr/share/gkrellm2/themes**. User GKrellM configuration information and support files are kept in the **.gkrellm2** directory. Here you will find subdirectories for themes and user plug-ins. The **user_config** file holds user configuration settings, listing a monitor, its options, and its settings on each line.

GKrellM Server

You can use GKrellM to monitor hosts remotely using the GKrellM server, **gkrellmd**. You run the server on the system you wish to monitor, letting it allow remote systems to use **gkrellm** clients to gather and display its monitoring statistics. To run **gkrellm** as a client to gather and display information from another system running a **gkrellmd** server, you use the **-s** option and the server's host name. The server has to be configured to allow that remote host to connect. In the next example, a remote host connects to a **gkrellmd** server running on **turtle.mytrek.com** to display information about the turtle host:

```
gkrellm -s turtle.mytrek.com
```

Configuration for the GKrellM server is handled by the **/etd/gkrellmd.conf** configuration file. Here you can specify the hosts to monitor as well as global options such as the frequency of updates, the port to listen on, and the maximum number of simultaneous clients. Options are documented in detail. Check the Man page for **gkrellmd** for a complete listing.

KDE Task Manager and Performance Monitor (KSysguard)

The KDE Task Manager and Performance Monitor, KSysguard, is accessible from the KDE desktop. This tool allows you to monitor the performance of your own system as well as remote systems. KSysguard can provide simple values or detailed tables for various parameters. A System Load panel provides graphical information about CPU and memory usage, and a Process Table lists current processes, using a tree format to show dependencies. You can design your own monitoring panels with worksheets, showing different types of values you want to display and the form you want to display them in, such as a bar graph or a digital meter. The Sensor Browser pane is an expandable tree of sensors for information like CPU System Load or Memory's Used Memory. There is a top entry for each host you are connected to, including your own, localhost. To design your own monitor, create a worksheet and drag and drop a sensor onto it.

Grand Unified Bootloader (GRUB)

The Grand Unified Bootloader (GRUB) is a multiboot boot loader used for most Linux distributions. With GRUB, users can select operating systems to run from a menu interface displayed when a system boots up. Use arrow keys to move to an entry and press ENTER. Type **e** to edit a command, letting you change kernel arguments or specify a different kernel. The **c** command places you in a command line interface. Provided your system BIOS supports very large drives, GRUB can boot from anywhere on them. Linux and Unix operating systems are known as multiboot operating systems and take arguments passed to them at boot time. Check the GRUB Man page for GRUB options. GRUB is a GNU project with its home page at **www.gnu.org/software/grub** and Wiki at **grub.enbug.org**.

NOTE *Some distributions provide boot configuration tools to select your default system or kernel as well as set the timeout limit).*

There are two versions of Grub available, Legacy Grub and Grub2. Legacy Grub is used on most distributions and is described in detail here. Grub2 is still under development, though available. Eventually Grub2 will replace Grub Legacy which is no longer being developed.

Officially, the Grub configuration settings are held in the **/boot/grub/menu.lst** file (Debian and Ubuntu). On Red Hat, Fedora, and similar distributions, GRUB configuration is held in the **/boot/grub/grub.conf** file (**menu.lst** is a link to this file). You only need to make your entries in the grub conifguation file and GRUB will automatically read them when you reboot. There are several options you can set, such as the timeout period and the background image to use. Check the GRUB info documentation for a detailed description, **info grub**. You can specify a system to boot by creating a title entry for it, beginning with the term **title**. You then have to specify where the operating system kernel or program is located, which hard drive to use, and what partition on that hard drive. This information is listed in parentheses following the **root** option. Numbering starts from 0, not 1, and hard drives are indicated with an **hd** prefix, whether they are IDE or SCSI hard drives. Thus **root(hd0,2)** references the first hard drive (**hda**) and the third partition on that hard drive (**hda3**). For Linux systems, you will also have to use the **kernel** option to indicate the kernel program to run, using the full pathname and any options the kernel may need. The RAM disk is indicated by the **initrd** option.

```
title Fedora Linux (2.6.21-1)
      root (hd0,2)
      kernel /boot/vmlinuz-2.6.21-1 ro root=/dev/hda3
      initrd /boot/initrd-2.6.21-1.img
```

The kernel option specifies the kernel to run. The kernel is located in the **/boot** directory and has the name **vmlinuz** with the kernel version number. You can have several kernels in the **/boot** directory and use GRUB to choose the one to use. After the kernel program you specify any options you want for the kernel. This includes an **ro** option, which initially starts the kernel as read only. The root option is used to specify the device on which your system was installed, your root directory. In the preceding example the system was installed on device **/dev/hda3**. If the root directory is on a logical volume, where it is normally installed, the device name would be something like **/dev/VolGroup00/LogVol01**.

If you installed the standard workstation configuration, your root directory will be installed on a logical volume. Your root option references the logical volume, specifying the volume group and the logical volume, as shown here.

```
kernel /vmlinuz-2.6.21-1 ro root=/dev/VolGroup00/LogVol00
```

Grub2 uses a different syntax, relying on a Bash shell like format. Configuration is held in a **grub.cfg** file. Menu items are defined with a **menuentry** command which hold menu configuration options. The commands are the same as in Grub Legacy. Of special note is that the partition numbering in Grub2 starts from 1, not 0, like in Grub Legacy. Hard disks still start from 0. So the third partition on the first hard disk is **hd0,3**. The following entry shows the previous **title** example using a Grub2 syntax.

```
menuentry "Fedora Linux (2.6.21-1)" {
     set root=(hd0,3)
     linux /boot/vmlinuz ro root=/dev/hda3
     initrd /boot/initrd-2.6.21-1.img
    }
```

For another operating system such as Windows, you use the **rootnoverify** option to specify where Windows is installed. This option instructs GRUB not to try to mount the partition. Use the **chainloader+1** option to allow GRUB to access it. The **chainloader** option tells GRUB to use another boot program for that operating system. The number indicates the sector on the partition where the boot program is located—for example, **+1** indicates the first sector.

```
title Windows XP
      rootnoverify (hd0,0)
      chainloader +1
```

Windows systems will all want to boot from the first partition on the first disk. This becomes a problem if you want to install several versions of Windows on different partitions or install Windows on a partition other than the first one. For Windows partitions on the same disk, GRUB lets you work around this by letting you hide other partitions in line, and then unhiding the one you want, making it appear to be the first partition. In this example, the first partition is hidden, and the second is unhidden. This assumes there is a Windows system on the second partition on the first hard drive (**hd0,1**). Now that the first partition is hidden, the second one appears as the first partition:

```
hide (hd0,0)
unhide (hd0,1)
rootnoverify (hd0,1)
```

For systems that have multiple hard drives, you may have Windows installed on a drive other than the first hard drive. GRUB numbers hard drives from 0, with **hd1** referencing the second hard drive, and **hd0** referencing the first hard drive. Windows will always want to boot from the first partition on the first hard drive. For a version of Windows installed on a hard drive other than the first one, GRUB lets you work around it by letting you renumber your drives with the **map** command. The first drive can be renumbered as another drive, and that drive can then ten be remapped as the first drive. In this example, the first drive is remapped as the second drive, and the second drive is mapped as the first drive. This example assumes

there is a Windows system on the first partition on the second hard drive (**hd1,0**). Once the first drive is remapped as the second one, the second drive can operate as the first drive. However, the **chainloader** operation still detects the actual location of that Windows OS on the second hard drive, **(hd1,0)+1**, in this example on the first partition. GRUB will then boot the Windows partition on the second hard drive, as if it where located on a first hard drive.

```
map (hd0) (hd1)
map (hd1) (hd0)
chainloader (hd1,0)+1
```

TIP *If you have problems booting to Linux, and you can fix the issue by editing the Grub configuration file (as by changing hard drive numbers), you can boot up with your CD/DVD Linux install disk and type* ***linux rescue*** *at the boot prompt. Follow the prompts to boot up your system with the command line interface. Issue the* ***chroot /mnt/sysimage*** *command. You can then change to the* ***/boot/grub*** *directory and edit your Grub configuration file with an editor like* ***vi****.*

A sample Grub configurations file (**menu.lst**) follows with entries for both Linux and Windows. Fedora kernel examples are used. Notice that kernel parameters are listed in the **kernel** option as arguments to the kernel. The root directory is installed on a logical volume, **/dev/VolGroup00/LogVol01**.

```
# grub.conf generated by anaconda
#
# Note that you do not have to rerun grub after making changes to this file
# NOTICE:  You have a /boot partition.  This means that
#          all kernel and initrd paths are relative to /boot/, eg.
#          root (hd0,1)
#          kernel /vmlinuz-version ro root=/dev/VolGroup00/LogVol01
#          initrd /initrd-version.img
#boot=/dev/sda
default=1
timeout=5
splashimage=(hd0,1)/grub/splash.xpm.gz
hiddenmenu
title Fedora (2.6.21-1)
     root (hd0,1)
     kernel /vmlinuz-2.6.21-1 ro root=/dev/VolGroup00/LogVol01
     initrd /initrd-2.6.21-1.img
title Windows XP
     rootnoverify (hd0,0)
     chainloader +1
```

The **title** entries from the previous example using Grub2 syntax would look like the following. Keep in mind that partition numbering starts from 1 in Grub2, not 0. The first partition on the first drive is **hd0,1**.

```
menuentry "Fedora Linux (2.6.21-1)" {
     set root=(hd0,3)
     linux /boot/vmlinuz ro root=/dev/hda3
     initrd /boot/initrd-2.6.21-1.img
    }
```

```
menuentry "Windows XP" {
     rootnoverify (hd0,1)
     chainloader +1
    }
```

NOTE *Some Linux distributions provide the older Linux Loader (LILO) as its boot manager. It performs the same kinds of tasks as GRUB. You can modify your LILO configuration either by using an administration tool like Boot Manager (LILO-config) or by editing the* ***/etc/lilo.conf*** *configuration file directly. You can also configure LILO with the KDE LILO configuration tool (lilo-config).*

CHAPTER 28

Managing Users

As a system administrator, you must manage the users of your system. You can add or remove users, as well as add and remove groups, and you can modify access rights and permissions for both users and groups. You also have access to system initialization files you can use to configure all user shells, and you have control over the default initialization files copied into a user account when it is first created. You can decide how new user accounts should be configured initially by configuring these files.

GUI User Managment Tools: users-admin and KUser

User's can be more easily managed using a GUI User management tool like GNOME's user-admin and KDE's KUser. GNOME's user-admin tool is part of GNOME's system tools package. Though some distributions like Red Hat still use their own custom designed tools, other distributions like Ubuntu are making use of GNOME's user-admin tool. The KUser tool has always been available on all distributions with the KDE desktop.

GNOME's **users-admin** tool provides a simple interface for adding, modifiying, and removing users and groups. It opens up with a Users Settings window listing users with their full name, login name, and home directory. An Add User buton to the side will open a New User Accounts window with Account, User Privileges, and Advanced panels. On the Account panel you can enter the login name and password. A profile pop-up menu lets you specify whether to make the user a normal user or an administrator. You can also add in contact information.

The User Privileges menu lets you control what a user can do, most importantly whether to grant administrative access. The Advanced panel lets you specify the user account settings like the home directory, the login shell to use, and what group to belong to. Default entries are already set up for you.

To later change settings, select the User in the User Settings window and click the Properties button. A three panel Account Properties window opens with the same Account, Privileges, and Advanced panels. To delete a user, select it and click the Delete button.

To manage groups, click on the Groups button. This opens a Group Settings window that lists all groups. To add users to a group, select it and click Properties. In the the Properties window, users will be listed and you can select the ones you want to add. To add a new group, click on the Add Group button to open a New Group window where you can specify the group name, its id, and select users to add to the group.

On KDE, the KDE User Manager (KUser) lets you manage both users and groups. The KDE User Manger window displays two panels, one for users and the other for groups. The Users panel lists user's user login, full name, home directory, and login shell, along with either user ID. On the toolbar there are Add, Edit, and Delete buttons for both users and groups, as well as Users and Groups menus with corresponding entries. Initially all system users and groups will also be displayed. Select Hide system users/groups from the Settings menu to display just normal users and groups.

When you add a new user, you are first prompted for the user name, then the KUser properties window opens with panels for User Info, Password Properties, and Groups. Default entries are already set. You really only need to enter the user's full name and the password (click Set Password). The Password panel lets you set expiration options, and the Groups panel lets you select groups you want the user to belong to.

You can set default configurations for new users with the KUser configuration window, accessible from the Settings menu. Here you can set a password encryption and expiration policies along with home directory and login shell defaults. LDAP support is also provided.

TIP *Every file is owned by a user who can control access to it. System files are owned by the root user and accessible by the root only. Services like FTP are an exception to this rule. Though accessible by the root, a service's files are owned by their own special user. For example, FTP files are owned by an* ***ftp*** *user. This provides users with access to a service's files without also having root user access.*

User Configuration Files

Any utility to manage a user, such as GNOME's users-adminor KDE's KUser, makes use of certain default files, called *configuration files,* and directories to set up the new account. A set of pathnames is used to locate these default files or to indicate where to create certain user directories. For example, **/etc/skel** holds initialization files for a new user. A new user's home directory is created in the **/home** directory. Table 28-1 has a list of the pathnames.

Directory and Files	Description
/home	The user's own home directory
/etc/skel	The default initialization files for the login shell, such as **.bash_profile**, **.bashrc**, and **.bash_logout**; includes many user setup directories and files such as **.kde** for KDE and **Desktop** for GNOME
/etc/shells	The login shells, such as BASH or TCSH
/etc/passwd	The password for a user
/etc/group	The group to which the user belongs
/etc/shadow	Encrypted password file
/etc/gshadow	Encrypted password file for groups
/etc/login.defs	Default login definitions for users

TABLE 28-1 Paths for User Configuration Files

TIP *You can find out which users are currently logged in with the* **w** *or* **who** *command. The* **w** *command displays detailed information about each connected user, such as from where they logged in and how long they have been inactive, and the date and time of login. The* **who** *command provides less detailed data.*

The Password Files

A user gains access to an account by providing a correct login and password. The system maintains passwords in password files, along with login information like the username and ID. Tools like the **passwd** command let users change their passwords by modifying these files; **/etc/passwd** is the file that traditionally held user passwords, though in encrypted form. However, all users are allowed to read the **/etc/passwd** file, which allows access by users to the encrypted passwords. For better security, password entries are now kept in the **/etc/shadow** file, which is restricted to the root user.

/etc/passwd

When you add a user, an entry for that user is made in the **/etc/passwd** file, commonly known as the *password file.* Each entry takes up one line that has several fields separated by colons. The fields are as follows:

- **Username** Login name of the user
- **Password** Encrypted password for the user's account
- **User ID** Unique number assigned by the system
- **Group ID** Number used to identify the group to which the user belongs
- **Comment** Any user information, such as the user's full name
- **Home directory** The user's home directory
- **Login shell** Shell to run when the user logs in; this is the default shell, usually **/bin/bash**

Depending on whether or not you are using shadow passwords, the password field (the second field) will be either an **x** or an encrypted form of the user's password. Linux implements shadow passwords by default, so these entries should have an **x** for their passwords. The following is an example of an **/etc/passwd** entry. For such entries, you must use the **passwd** command to create a password. Notice also that user IDs in this particular system start at 500 and increment by one. The group given is not the generic User, but a group consisting uniquely of that user. For example, the **dylan** user belongs to a group named **Dylan**, not to the generic **User** group.

```
dylan:x:500:500:Dylan:/home/dylan:/bin/bash
chris:x:501:501:Chris:/home/chris:/bin/bash
```

TIP *If you turn off shadow password support, entries in your* **passwd** *file will display encrypted passwords. Because any user can read the* **/etc/passwd** *file, intruders can access and possibly crack the encrypted passwords.*

TIP *Although it is technically possible to edit entries in the **/etc/passwd** file directly, it is not recommended. In particular, deleting an entry does not remove any other information, permissions, and data associated with a user, which opens a possible security breach whereby an intruder could take over the deleted user's ID or disk space.*

/etc/shadow and /etc/gshadow

The **/etc/passwd** file is a simple text file and is vulnerable to security breaches. Anyone who gains access to the **/etc/password** file might be able to decipher or crack the encrypted passwords through a brute-force crack. The shadow suite of applications implements a greater level of security. These include versions of **`useradd`**, **`groupadd`**, and their corresponding update and delete programs. Most other user configuration tools support shadow security measures. With shadow security, passwords are no longer kept in the **/etc/password** file. Instead, passwords are kept in a separate file called **/etc/shadow**. Access is restricted to the root user.

The following example shows the **/etc/passwd** entry for a user.

```
chris:x:501:501:Chris:/home/chris:/bin/bash
```

A corresponding password file, called **/etc/gshadow,** is also maintained for groups that require passwords.

Password Tools

To change any particular field for a given user, you should use the user management tools provided, such as the **`passwd`** command, **`adduser`**, **`usermod`**, **`useradd`**, and **`chage`**, discussed in this chapter. The **`passwd`** command lets you change the password only. Other tools not only make entries in the **/etc/passwd** file, but also create the home directory for the user and install initialization files in the user's home directory.

These tools also let you control users' access to their accounts. You can set expiration dates for users or lock them out of their accounts. Users locked out of their accounts will have a their password in the **/etc/shadow** file prefixed by the invalid string, **`!!`**. Unlocking the account removes this prefix.

Managing User Environments

Each time a user logs in, two profile scripts are executed, a system profile script that is the same for every user, and a user login profile script that can be customized to each user's needs. When the user logs out, a user logout script is run. In addition, each time a shell is generated, including the login shell, a user shell script is run. There are different kinds of scripts used for different shells. The default shell commonly used is the BASH shell. As an alternative, users can use different shells such as TCSH or the Z shell.

Profile Scripts

For the BASH shell, each user has his or her own BASH login profile script named **.bash_profile** in the user's home directory. The system profile script is located in the **/etc** directory and named **profile** with no preceding period. The BASH shell user shell script is called **.bashrc**. The **.bashrc** file also runs the **/etc/bashrc** file to implement any global definitions

such as the **PS1** and **TERM** variables. The **/etc/bashrc** file also executes any specialized initialization file in the **/etc/profile.d** directory, such as those used for KDE and GNOME. The **.bash_profile** file runs the **.bashrc** file, and through it, the **/etc/bashrc** file, implementing global definitions.

As a superuser, you can edit any of these profile or shell scripts and put in any commands you want executed for each user when that user logs in. For example, you may want to define a default path for commands, in case the user has not done so. Or you may want to notify the user of recent system news or account changes.

/etc/skel

When you first add a user to the system, you must provide the user with skeleton versions of their login, shell, and logout initialization files. For the BASH shell, these are the **.bash_profile**, **.bashrc**, and **.bash_logout** files. The **useradd** command and other user management tools add these files automatically, copying any files in the directory **/etc/skel** to the user's new home directory. The **/etc/skel** directory contains a skeleton initialization file for the **.bash_profile**, **.bashrc**, and **.bash_logout** files or, if you are using the TCSH shell as your login shell, the **.login**, **.tcshrc,** and **.logout** files. The **/etc/skel** directory also contains default files and directories for your desktops. These include a **.screenrc** file for the X Window System, a **.kde** directory for the KDE desktop, and a **Desktop** directory that contains default configuration files for the GNOME desktop.

As a superuser, you can configure the **.bash_profile** or **.bashrc** file in the **/etc/skel** directory any way you want. Usually, basic system variable assignments are included that define pathnames for commands and command aliases. The **PATH** and **BASH_ENV** variables are defined in **.bash_profile**. Once users have their own **.bash_profile** or **.bashrc** file, they can redefine variables or add new commands as they choose.

/etc/login.defs

Systemwide values used by user and group creation utilities such as **useradd** and **usergroup** are kept in the **/etc/login.defs** file. Here you will find the range of possible user and group IDs listed. **UID_MIN** holds the minimum number for user IDs and **UID_MAX** the maximum number. Various password options control password controls—such as **PASS_MIN_LEN**, which determines the minimum number of characters allowable in a password. Options such as **CREATE_HOME** can be set to tell user tools like **useradd** to create home directories for new accounts by default. Samples of these entries are shown here:

```
MAIL_DIR /var/spool/mail
PASS_MIN_LEN        5
CREATE_HOME yes
```

/etc/login.access

You can control user login access by remote users to your system with the **/etc/login.access** file. The file consists of entries listing users, whether they are allowed access, and from where they can access the system. A record in this file consists of three colon-delimited fields: a plus (**+**) or minus (**-**) sign indicating whether users are allowed access, user login names allowed access, and the remote system (host) or terminal (tty device) from which

they are trying to log in. The following enables the user **chris** to access the system from the **rabbit.mytrek.com** remote system:

```
+:chris:rabbit.mytrek.com
```

You can list more than one user or location or use the **ALL** option in place of either users or locations to allow access by all users and locations. The **ALL** option can be qualified with the **EXCEPT** option to allow access by all users except certain specified ones. The following entry allows any valid user to log in to the system using the console, except for the users **larisa** and **aleina**:

```
+:ALL EXCEPT larisa aleina:console
```

Other access control files are used to control access for specific services, such as the **hosts.deny** and **hosts.allows** files used with the **tcpd** daemon for **xinetd**-supported servers.

Controlling User Passwords

Once you have created a user account, you can control the user's access to it. The **passwd** tool lets you lock and unlock a user's account. You use the **passwd** command with the **-l** option to lock an account, invalidating its password, and you use the **-u** option to unlock it.

You can also force a user to change his or her password at given intervals by setting an expiration date for that password. The **chage** command let you specify an expiration limit for a user's password. A user can be required to change his or her password every month, every week, or at a given date. Once the password expires, the user is prompted to enter a new one. You can issue a warning beforehand, telling the user how much time is left before the password expires. If there is an account that you want to close, you can permanently expire a password. You can even shut down accounts that are inactive too long. In the next example, the password for the **chris** account will stay valid for only seven days. The **-M** option with the number of days sets the maximum time that a password can be valid.

```
chage -M 7  chris
```

To set a particular date for the account to expire, use the **-E** option with the date specified mm/dd/yyyy.

```
chage -E 07/30/2003  chris
```

To find out what the current expiration settings are for a given account, use the **-l** option.

```
chage -l chris
```

You can also combine your options into one command.

```
chage -M 7 -E 07/30/2003  chris
```

A listing of the **chage** options appears in Table 28-2.

Option	Description
`-m`	Minimum number of days a user must go before being able to change his password
`-M`	Maximum number of days a user can go without changing her password
`-d`	The last day the password was changed
`-E`	Specific expiration date for a password, date in format in yyyy-mm-dd or in commonly used format like mm/dd/yyyy
`-I`	Allowable account inactivity period (in days), after which password will expire
`-W`	Warning period, number of days before expiration when the user will be sent a warning message
`-l`	Display current password expiration controls

TABLE 28-2 Options for the **chage** Command

Adding and Removing Users with useradd, usermod, and userdel

Linux also provides the **useradd**, **usermod**, and **userdel** commands to manage user accounts. All these commands take in their information as options on the command line. If an option is not specified, they use predetermined default values. These are command line operations. To use them on your desktop you first need to open a terminal window (right-click the desktop and select Open Terminal), and then enter the commands at the shell prompt.

If you are using a desktop interface, you should use GUI tools to manage user accounts. Each Linux distribution usually provides a tool to manage users. In addition you can use the K Desktop KUser tool or the GNOME System Tools User Settings. See Table 28-3 for a listing of user management tools.

Tool	Description
KUser	K Desktop tool for adding, removing, and modifying users and groups
GNOME Users settings	GNOME desktop tool for adding, removing, and modifying users and groups
useradd *username options*	Adds a user
userdel *username*	Deletes a user
usermod *username options*	Modifies a user properties
groupadd *groupname options*	Adds a group
groupdel *groupname*	Deletes a group
groupmod *groupname options*	Modifies a group name

TABLE 28-3 User and Group Management Tools

useradd

With the **useradd** command, you enter values as options on the command line, such as the name of a user, to create a user account. It then creates a new login and directory for that name using all the default features for a new account.

```
# useradd chris
```

The **useradd** utility first checks the **/etc/login.defs** file for default values for creating a new account. For those defaults not defined in the **/etc/login.defs** file, **useradd** supplies its own. You can display these defaults using the **useradd** command with the **-D** option. The default values include the group name, the user ID, the home directory, the **skel** directory, and the login shell. Values the user enters on the command line will override corresponding defaults. The group name is the name of the group in which the new account is placed. By default, this is **other**, which means the new account belongs to no group. The user ID is a number identifying the user account. The **skel** directory is the system directory that holds copies of initialization files. These initialization files are copied into the user's new home directory when it is created. The login shell is the pathname for the particular shell the user plans to use.

The **useradd** command has options that correspond to each default value. Table 28-4 holds a list of all the options you can use with the **useradd** command. You can use specific

Option	Description
-d *dir*	Sets the home directory of the new user.
-D	Displays defaults for all settings. Can also be used to reset default settings for the home directory (**-b**), group (**-g**), shell (**-s**), expiration date (**-e**), and password expirations (**-f**).
-e *mm/dd/yy*	Sets an expiration date for the account (none, by default). Specified as month/day/year.
-f *days*	Sets the number of days an account remains active after its password expires.
-g *group*	Sets a group.
-m	Creates user's home directory, if it does not exist.
-m -k *skl-dir*	Sets the skeleton directory that holds skeleton files, such as **.profile** files, which are copied to the user's home directory automatically when it is created; the default is **/etc/skel**.
-M	Does not create user's home directory.
-p *password*	Supplies an encrypted password (crypt or MD5). With no argument, the account is immediately disabled.
-s *shell*	Sets the login shell of the new user. This is **/bin/bash** by default, the BASH shell.
-u *userid*	Sets the user ID of the new user. The default is the increment of the highest number used so far.

TABLE 28-4 Options for useradd and usermod

values in place of any of these defaults when creating a particular account. The login is inaccessible until you do. In the next example, the group name for the **chris** account is set to **intro1** and the user ID is set to 578:

```
# useradd chris -g intro1 -u 578
```

Once you add a new user login, you need to give the new login a password. Password entries are placed in the **/etc/passwd** and **/etc/shadow** files. Use the **passwd** command to create a new password for the user, as shown here. The password you enter will not appear on your screen. You will be prompted to repeat the password. A message will then be issued indicating that the password was successfully changed.

```
# passwd chris
Changing password for user chris
New UNIX password:
Retype new UNIX password:
passwd: all authentication tokens updated successfully
#
```

usermod

The **usermod** command enables you to change the values for any of these features. You can change the home directory or the user ID. You can even change the username for the account. The **usermod** command takes the same options as **useradd**, listed previously in Table 28-4.

userdel

When you want to remove a user from the system, you can use the **userdel** command to delete the user's login. With the **-r** option, the user's home directory will also be removed. In the next example, the user **chris** is removed from the system:

```
# userdel -r chris
```

Managing Groups

You can manage groups using either shell commands or GUI utilities. Groups are an effective way to manage access and permissions, letting you control several users with just their group name.

/etc/group and /etc/gshadow

The system file that holds group entries is called **/etc/group**. The file consists of group records, with one record per line and its fields separated by colons. A group record has four fields: a group name, a password, its ID, and the users who are part of this group. The Password field can be left blank. The fields for a group record are as follows:

- **Group name** The name of the group, which must be unique
- **Password** With shadow security implemented, this field is an **x**, with the password indicated in the **/etc/gshadow** file

- **Group ID** The number assigned by the system to identify this group
- **Users** The list of users that belong to the group, separated by commas

Here is an example of an entry in an **/etc/group** file. The group is called **engines**, the password is managed by shadow security, the group ID is 100, and the users who are part of this group are **chris**, **robert**, **valerie**, and **aleina**:

```
engines:x:100:chris,robert,valerie,aleina
```

As in the case of the **/etc/passwd** file, it is best to change group entries using a group management utility like **groupmod** or **groupadd**. All users have read access to the **/etc/group** file. With shadow security, secure group data such as passwords are kept in the **/etc/gshadow** file, to which only the root user has access.

User Private Groups

A new user can be assigned to a special group set up for just that user and given the user's name. Thus the new user **dylan** is given a default group also called **dylan**. The group **dylan** will also show up in the listing of groups. This method of assigning default user groups is called the User Private Group (UPG) scheme. The supplementary groups are additional groups that the user may want to belong to. Traditionally, users were all assigned to one group named **users** that subjected all users to the group permission controls for the **users** group. With UPG, each user has its own group, with its own group permissions.

Group Directories

As with users, you can create a home directory for a group. To do so, you simply create a directory for the group in the **/home** directory and change its home group to that group and allow access by any member of the group. The following example creates a directory called **engines** and changes its group to the **engines** group:

```
mkdir /home/engines
chgrp engines /home/engines
```

Then the read, write, and execute permissions for the group level should be set with the **chmod** command, discussed later in this chapter:

```
chmod g+rwx /home/engines
```

Any member of the **engines** group can now access the **/home/engines** directory and any shared files placed therein. This directory becomes a shared directory for the group. You can, in fact, use the same procedure to make other shared directories at any location on the file system.

Files within the shared directory should also have their permissions set to allow access by other users in the group. When a user places a file in a shared directory, the user needs to set the permissions on that file to allow other members of the group to access it. A read permission will let others display it, write lets them change it, and execute lets them run it (used for scripts and programs). The following example first changes the group for the

mymodel file to **engines**. Then it copies the **mymodel** file to the **/home/engines** directory and sets the group read and write permission for the **engines** group:

```
$ chgrp engines mymodel
$ cp mymodel /home/engines
$ chmod g+rw /home/engines/mymodel
```

Managing Groups Using groupadd, groupmod, and groupdel

You can also manage groups with the **groupadd**, **groupmod**, and **groupdel** commands. These command line operations let you quickly manage a group from a terminal window.

groupadd and groupdel

With the **groupadd** command, you can create new groups. When you add a group to the system, the system places the group's name in the **/etc/group** file and gives it a group ID number. If shadow security is in place, changes are made to the **/etc/gshadow** file. The **groupadd** command only creates the group category. You need to add users to the group individually. In the following example, the **groupadd** command creates the **engines** group:

```
# groupadd engines
```

You can delete a group with the **groupdel** command. In the next example, the **engines** group is deleted:

```
# groupdel engines
```

groupmod

You can change the name of a group or its ID using the **groupmod** command. Enter **groupmod -g** with the new ID number and the group name. To change the name of a group, you use the **-n** option. Enter **groupmod -n** with the new name of the group, followed by the current name. In the next example, the **engines** group has its name changed to **trains**:

```
# groupmod -n trains engines
```

Controlling Access to Directories and Files: chmod

Each file and directory in Linux contains a set of permissions that determine who can access them and how. You set these permissions to limit access in one of three ways: you can restrict access to yourself alone, you can allow users in a predesignated group to have access, or you can permit anyone on your system to have access. You can also control how a given file or directory is accessed.

NOTE *See Chapter 17 to learn how to use SELinux to set permissions on users and files.*

Permissions

A file or directory may have read, write, and execute permissions. When a file is created, it is automatically given read and write permissions for the owner, enabling you to display and modify the file. You may change these permissions to any combination you want. A file can also have read-only permission, preventing any modifications.

TIP *From GNOME and KDE you can change permissions easily by right-clicking on a file or directory icon and selecting Properties. On the Permissions panel you will see options for setting Owner, Group, and Other permissions.*

Permission Categories

Three different categories of users can have access to a file or directory: the owner, the group, and all others not belonging to that group. The owner is the user who created the file. Any file you create, you own. You can also permit a group to have access to a file. Often, users are collected into groups. For example, all the users for a given class or project can be formed into a group by the system administrator. A user can grant access to a file to the members of a designated group. Finally, you can also open up access to a file to all other users on the system. In this case, every user not part of the file's group can have access to that file. In this sense, every other user on the system makes up the "others" category. If you want to give the same access to all users on your system, you set the same permissions for both the group and the others. That way, you include both members of the group (group permission) and all those users who are not members (others permission).

Read, Write, Execute Permissions

Each category has its own set of read, write, and execute permissions. The first set controls the user's own access to his or her files—the owner access. The second set controls the access of the group to a user's files. The third set controls the access of all other users to the user's files. The three sets of read, write, and execute permissions for the three categories—owner, group, and other—make a total of nine types of permissions.

The **ls** command with the **-l** option displays detailed information about the file, including the permissions. In the following example, the first set of characters on the left is a list of the permissions set for the **mydata** file:

```
$ ls -l mydata
-rw-r--r-- 1 chris weather 207 Feb 20 11:55 mydata
```

An empty permission is represented by a dash, **-**. The read permission is represented by **r**, write by **w**, and execute by **x**. Notice there are ten positions. The first character indicates the file type. In a general sense, a directory can be considered a type of file. If the first character is a dash, a file is being listed. If the first character is **d**, information about a directory is being displayed.

The next nine characters are arranged according to the different user categories. The first set of three characters is the owner's set of permissions for the file. The second set of three characters is the group's set of permissions for the file. The last set of three characters is the other users' set of permissions for the file.

Permissions on GNOME

On GNOME, you can set a directory or file permission using the Permissions panel in its Properties window. Right-click the file or directory entry in the file manager window and select Properties. Then select the Permissions panel. Here you will find pop-up menus for Read, Write, and Execute along with rows for Owner, Group, and Other. You can set owner permissions as Read Only or Read And Write. For the group and others, you can also set the None option, denying access. The group name expands to a pop-up menu listing different groups; select one to change the file's group. If you want to execute this as an application

(say, a shell script) check the Allow Executing File As Program entry. This has the effect of setting the execute permission.

The Permissions panel for directories operates much the same way, but it includes two access entries, Folder Access and File Access. The Folder Access entry controls access to the folder with options for List Files Only, Access Files, and Create And Delete Files. These correspond to the read, read and execute, and read/write/execute permissions given to directories. The File Access entry lets you set permissions for all those files in the directory. They are the same as for files: for the owner, Read or Read and Write; for the group and others, the entry adds a None option to deny access. To set the permissions for all the files in the directory accordingly (not just the folder), click the Apply Permissions To Enclosed Files button.

Permissions on KDE

On KDE, you can set a directory or file permission using the Permissions panel in its Properties window. Right-click the file or directory entry in the file manager window and select Properties. Then select the Permissions panel. Here you will find pop-up menus for Owner, Group, and Others. Options include Can Read, Can Read and Write, and Forbidden. For more refined access, click on the Advanced Permissions button to display a table for checking read, write, and execute access (**r**, **w**, **x**) for owner, group, and others. You can also set the sticky bit and user and group ID permissions. The Add Entry button lets you set up ACL access, specifying certain users or groups that can or cannot have access to the file.

Directories have slightly different options: Can View Content and Can View and Modify Content which are the read and write permissions. You have the option to apply changes to all subdirectories and the files in them. Clicking on the Advanced Permissions button displays the same read, write, and execute table for owner, group, and others. Click a table entry to toggle a permission on or off. The selected permissions are shown in the Effective column. Use the Add Entry button to add ACL entries to control access by specific users and groups.

chmod

You use the **chmod** command to change different permission configurations. **chmod** takes two lists as its arguments: permission changes and filenames. You can specify the list of permissions in two different ways. One way uses permission symbols and is referred to as the *symbolic method.* The other uses what is known as a "binary mask" and is referred to as either the *absolute* or the *relative method*. Table 28-5 lists options for the **chmod** command.

NOTE *When a program is owned by the root, setting the user ID permission will give the user the ability to execute the program with root permissions. This can be a serious security risk for any program that can effect changes—such as* **rm***, which removes files.*

Ownership

Files and directories belong to both an owner and a group. A group usually consists of a collection of users, all belonging to the same group. In the following example, the **mydata** file is owned by the user **robert** and belongs to the group **weather**:

```
-rw-r--r-- 1 robert weather 207 Feb 20 11:55 mydata
```

Command or Option	Execution
chmod	Changes the permission of a file or directory.
Options	
+	Adds a permission.
-	Removes a permission.
=	Assigns entire set of permissions.
r	Sets read permission for a file or directory. A file can be displayed or printed. A directory can have the list of its files displayed.
w	Sets write permission for a file or directory. A file can be edited or erased. A directory can be removed.
x	Sets execute permission for a file or directory. If the file is a shell script, it can be executed as a program. A directory can be changed to and entered.
u	Sets permissions for the user who created and owns the file or directory.
g	Sets permissions for group access to a file or directory.
o	Sets permissions for access to a file or directory by all other users on the system.
a	Sets permissions for access by the owner, group, and all other users.
s	Sets User ID and Group ID permission; program owned by owner and group.
t	Sets sticky bit permission; program remains in memory.
Commands	
chgrp *groupname filenames*	Changes the group for a file or files.
chown *user-name filenames*	Changes the owner of a file or files.
ls -l *filename*	Lists a filename with its permissions displayed.
ls -ld *directory*	Lists a directory name with its permissions displayed.
ls -l	Lists all files in a directory with its permissions displayed.

TABLE 28-5 File and Directory Permission Operations

A group can also consist of one user, normally the user who creates the file. Each user on the system, including the root user, is assigned his or her own group of which he or she is the only member, ensuring access only by that user. In the next example, the report file is owned by the **robert** user and belongs to that user's single user group, **robert**:

```
-rw-r--r-- 1 robert robert 305 Mar 17 12:01 report
```

The root user, the system administrator, owns most of the system files that also belong to the root group, of which only the root user is a member. Most administration files, like configuration files in the **/etc** directory, are owned by the root user and belong to the root group. Only the root user has permission to modify them, whereas normal users can read and, in the case of programs, also execute them. In the next example, the root user owns the **fstab** file in the **/etc** directory, which also belongs to the root user group.

```
-rw-r--r-- 1 root root 621 Apr 22 11:03 fstab
```

Certain directories and files located in the system directories are owned by a service, rather than the root user, because the services need to change those files directly. This is particularly true for services that interact with remote users, such as Internet servers. Most of these files are located in the **/var** directory. Here you will find files and directories managed by services like the Squid proxy server and the Domain Name Server (named). In this example, the Squid proxy server directory is owned by the **squid** user and belongs to the **squid** group:

```
drwxr-x--- 2 squid squid 4096 Jan 24 16:29 squid
```

Changing a File's Owner or Group: chown and chgrp

Although other users may be able to access a file, only the owner can change its permissions. If, however, you want to give some other user control over one of your file's permissions, you can change the owner of the file from yourself to the other user. The **chown** command transfers control over a file to another user. This command takes as its first argument the name of the other user. Following the username, you list the files you are transferring. In the next example, the user gives control of the **mydata** file to user **robert**:

```
$ chown robert mydata
$ ls -l mydata
-rw-r--r-- 1 robert weather 207 Feb 20 11:55 mydata
```

You can also, if you wish, change the group for a file, using the **chgrp** command. **chgrp** takes as its first argument the name of the new group for a file or files. Following the new group name, you list the files you want changed to that group. In the next example, the user changes the group name for **today** and **weekend** to the **forecast** group. The **ls -l** command then reflects the group change.

```
$ chgrp forecast today weekend
$ ls -l
-rw-rw-r-- 1 chris forecast 568 Feb 14 10:30 today
-rw-rw-r-- 1 chris forecast 308 Feb 17 12:40 weekend
```

You can combine the **chgrp** operation with the **chown** command by attaching a group to the new owner with a colon.

```
$ chown george:forecast tomorrow
-rw-rw-r-- 1 george forecast 568 Feb 14 10:30 tomorrow
```

Setting Permissions: Permission Symbols

The symbolic method of setting permissions uses the characters **r**, **w**, and **x** for read, write, and execute, respectively. Any of these permissions can be added or removed. The symbol to add a permission is the plus sign, **+**. The symbol to remove a permission is the minus sign, **-**. In the next example, the **chmod** command adds the execute permission and removes the write permission for the **mydata** file for all categories. The read permission is not changed.

```
$ chmod +x-w mydata
```

Permission symbols also specify each user category. The owner, group, and others categories are represented by the **u**, **g**, and **o** characters, respectively. Notice the owner category is represented by a **u** and can be thought of as the user. The symbol for a category is placed before plus and minus sign preceding the read, write, and execute permissions. If no category symbol is used, all categories are assumed, and the permissions specified are set for the user, group, and others. In the next example, the first **chmod** command sets the permissions for the group to read and write. The second **chmod** command sets permissions for other users to read. Notice no spaces are between the permission specifications and the category. The permissions list is simply one long phrase, with no spaces.

```
$ chmod g+rw mydata
$ chmod o+r mydata
```

A user may remove permissions as well as add them. In the next example, the read permission is set for other users, but the write and execute permissions are removed:

```
$ chmod o+r-wx mydata
```

Another permission character exists, **a**, which represents all the categories. The **a** character is the default. In the next example, the two commands are equivalent. The read permission is explicitly set with the **a** character denoting all types of users: other, group, and user.

```
$ chmod a+r mydata
$ chmod +r mydata
```

One of the most common permission operations is setting a file's executable permission. This is often done in the case of shell program files. The executable permission indicates a file contains executable instructions and can be directly run by the system. In the next example, the file **lsc** has its executable permission set and then executed:

```
$ chmod u+x lsc
$ lsc
main.c lib.c
$
```

Absolute Permissions: Binary Masks

Instead of the permission symbols in Table 28-5, many users find it more convenient to use the absolute method. The *absolute method* changes all the permissions at once, instead of specifying one or the other. It uses a *binary mask* that references all the permissions in each category.

The three categories, each with three permissions, conform to an octal binary format. Octal numbers have a base 8 structure. When translated into a binary number, each octal digit becomes three binary digits. A binary number is a set of 1 and 0 digits. Three octal digits in a number translate into three sets of three binary digits, which is nine altogether—and the exact number of permissions for a file.

You can use the octal digits as a mask to set the different file permissions. Each octal digit applies to one of the user categories. You can think of the digits matching up with the permission categories from left to right, beginning with the owner category. The first octal digit applies to the owner category, the second to the group, and the third to the others category. The actual octal digit you choose determines the read, write, and execute permissions for each category. At this point, you need to know how octal digits translate into their binary equivalents.

Calculating Octal Numbers

A simple way to calculate the octal number makes use of the fact that any number used for permissions will be a combination derived from adding in decimal terms the numbers 4, 2, and 1. Use 4 for read permission, 2 for write, and 1 for execute. The read, write, execute permission is simply the addition of 4 + 2 + 1 to get 7. The read and execute permission adds 4 and 1 to get 5. You can use this method to calculate the octal number for each category. To get 755, you would add 4 + 2 + 1 for the owner read, write, and execute permission, 4 + 1 for the group read and execute permission, and 4 + 1 again for the other read and execute permission.

Binary Masks

When dealing with a binary mask, you need to specify three digits for all three categories, as well as their permissions. This makes a binary mask less versatile than the permission symbols. To set the owner execute permission on and the write permission off for the **mydata** file and retain the read permission, you need to use the octal digit 5 (101). At the same time, you need to specify the digits for group and other users access. If these categories are to retain read access, you need the octal number 4 for each (100). This gives you three octal digits, 544, which translate into the binary digits 101 100 100.

```
$ chmod 544 mydata
```

Execute Permissions

One of the most common uses of the binary mask is to set the execute permission. You can create files that contain Linux commands, called *shell scripts*. To execute the commands in a shell script, you must first indicate the file is executable—that it contains commands the system can execute. You can do this in several ways, one of which is to set the executable permission on the shell script file. Suppose you just completed a shell script file and you need to give it executable permission to run it. You also want to retain read and write permission but deny any access by the group or other users. The octal digit 7 (111) will set all three permissions, including execute (you can also add 4-read, 2-write, and 1-execute to get 7). Using 0 for the group and other users denies them access. This gives you the digits 700, which are equivalent to the binary digits 111 000 000. In this example, the owner permission for the **myprog** file is set to include execute permission:

```
$ chmod 700 myprog
```

If you want others to be able to execute and read the file but not change it, you can set the read and execute permissions and turn off the write permission with the digit 5 (101). In this case, you use the octal digits 755, having the binary equivalent of 111 101 101.

```
$ chmod 755 myprog
```

Directory Permissions

You can also set permissions on directories. The read permission set on a directory allows the list of files in a directory to be displayed. The execute permission enables a user to change to that directory. The write permission enables a user to create and remove his or her files in that directory. If you allow other users to have write permission on a directory, they can add their own files to it. When you create a directory, it is automatically given read, write, and execute permission for the owner. You may list the files in that directory, change to it, and create files in it.

Like files, directories have sets of permissions for the owner, the group, and all other users. Often, you may want to allow other users to change to and list the files in one of your directories but not let them add their own files to it. In this case, you set read and execute permissions on the directory, but you don't set a write permission. This allows other users to change to the directory and list the files in it but not to create new files or to copy any of their files into it. The next example sets read and execute permissions for the group for the **thankyou** directory but removes the write permission. Members of the group may enter the **thankyou** directory and list the files there, but they may not create new ones.

```
$ chmod g+rx-w letters/thankyou
```

Just as with files, you can also use octal digits to set a directory permission. To set the same permissions as in the preceding example, you use the octal digits 750, which have the binary equivalents of 111 101 000.

```
$ chmod 750 letters/thankyou
```

Displaying Directory Permissions

The **ls** command with the **-l** option lists all files in a directory. To list only the information about the directory itself, add a **d** modifier. In the next example, **ls -ld** displays information about the **thankyou** directory. Notice the first character in the permissions list is **d**, indicating it is a directory:

```
$ ls -ld thankyou
drwxr-x--- 2 chris 512 Feb 10 04:30 thankyou
```

Parent Directory Permissions

If you have a file you want other users to have access to, you not only need to set permissions for that file, you also must make sure the permissions are set for the directory in which the file is located. To access your file, a user must first access the file's directory. The same applies to parents of directories. Although a directory may give permission to others to access it, if its parent directory denies access, the directory cannot be reached. Therefore, you must pay close attention to your directory tree. To provide access to a directory, all other directories above it in the directory tree must also be accessible to other users.

Ownership Permissions

In addition to the read/write/execute permissions, you can also set ownership permissions for executable programs. Normally, the user who runs a program owns it while it is running, even though the program file itself may be owned by another user. The Set User ID permission allows the original owner of the program to own it always, even while another user is running the program. For example, most software on the system is owned by the root user but is run by ordinary users. Some such software may have to modify files owned by the root. In this case, the ordinary user needs to run that program with the root retaining ownership so that the program can have the permissions to change those root-owned files. The Group ID permission works the same way, except for groups. Programs owned by a group retain ownership, even when run by users from another group. The program can then change the owner group's files. There is a potential security risk involved in that you are essentially giving a user some limited root-level access.

Ownership Permissions Using Symbols

To add both the User ID and Group ID permissions to a file, you use the **s** option. The following example adds the User ID permission to the **pppd** program, which is owned by the root user. When an ordinary user runs **pppd**, the root user retains ownership, allowing the **pppd** program to change root-owned files.

```
# chmod +s /usr/sbin/pppd
```

The Set User ID and Set Group ID permissions show up as an **s** in the execute position of the owner and group segments. Set User ID and Group ID are essentially variations of the execute permission, **x**. Read, write, and User ID permission are **rws** instead of just **rwx**.

```
# ls -l /usr/sbin/pppd
-rwsr-sr-x 1 root root 184412 Jan 24 22:48 /usr/sbin/pppd
```

Ownership Permissions Using the Binary Method

For the ownership permissions, you add another octal number to the beginning of the octal digits. The octal digit for User ID permission is 4 (100) and for Group ID, it is 2 (010) (use 6 to set both—110). The following example sets the User ID permission to the **pppd** program, along with read and execute permissions for the owner, group, and others:

```
# chmod 4555 /usr/sbin/pppd
```

Sticky Bit Permissions

One other special permission provides for greater security on directories, the *sticky bit.* Originally the sticky bit was used to keep a program in memory after it finished execution to increase efficiency. Current Linux systems ignore this feature. Instead, it is used for directories to protect files within them. Files in a directory with the sticky bit set can only be deleted or renamed by the root user or the owner of the directory.

Sticky Bit Permission Using Symbols

The sticky bit permission symbol is **t**. The sticky bit shows up as a **t** in the execute position of the other permissions. A program with read and execute permissions with the sticky bit has its permissions displayed as **r-t**.

```
# chmod +t /home/dylan/myreports
# ls -l /home/dylan/myreports
-rwxr-xr-t 1 root root 4096 /home/dylan/myreports
```

Sticky Bit Permission Using the Binary Method

As with ownership, for sticky bit permissions, you add another octal number to the beginning of the octal digits. The octal digit for the sticky bit is 1 (001). The following example sets the sticky bit for the **myreports** directory:

```
# chmod 1755 /home/dylan/myreports
```

The next example sets both the sticky bit and the User ID permission on the **newprogs** directory. The permission 5755 has the binary equivalent of 101 111 101 101:

```
# chmod 5755 /usr/bin/newprogs
# ls -l /usr/bin/newprogs
drwsr-xr-t 1 root root 4096  /usr/bin/newprogs
```

Permission Defaults: umask

Whenever you create a file or directory, it is given default permissions. You can display the current defaults or change them with the **umask** command. The permissions are displayed in binary or symbolic format as described in the following sections. The default permissions include any execute permissions that are applied to a directory. Execute permission for a file is turned off by default when you create it because standard data files do not use the executable permissions (to make a file executable like a script, you have to manually set its execute permission). To display the current default permissions, use the **umask** command with no arguments. The **-S** option uses the symbolic format.

```
$ umask -S
u=rwx,g=rx,o=rx
```

This default umask provides rw-r--r-- permission for standard files and adds execute permission for directories, rwxr-xr-x.

You can set a new default by specifying permissions in either symbolic or binary format. To specify the new permissions, use the **-S** option. The following example denies others read permission, while allowing user and group read access, which results in permissions of rwxr-x---:

```
$ umask -S  u=rwx,g=rx,o=
```

When you use the binary format, the mask is the inverse of the permissions you want to set. To set both the read and execute permission on and the write permission off, you use the octal number 2, a binary 010. To set all permissions on, you use an octal 0, a binary 000.

The following example shows the mask for the permission defaults rwx, rx, and rx (rw, r, and r for files):

```
$ umask
0022
```

To set the default to only deny all permissions for others, you use 0027, using the binary mask 0111 for the other permissions.

```
$ umask 0027
```

Disk Quotas

You can use disk quotas to control how much disk space a particular user makes use of on your system. On your Linux system, unused disk space is held as a common resource that each user can access as he or she needs it. As users create more files, they take the space they need from the pool of available disk space. In this sense, all the users are sharing a single resource of unused disk space. However, if one user were to use up all the remaining disk space, none of the other users would be able to create files or even run programs. To counter this problem, you can create disk quotas on particular users, limiting the amount of available disk space they can use.

Quota Tools

Quota checks can be implemented on the file system of a hard disk partition mounted on your system. The quotas are enabled using the **quotacheck** and **quotaon** programs. They are executed in the **/etc/rc.d/rc.sysinit** script, which is run whenever you start up your system. Each partition needs to be mounted with the quota options, **usrquota** or **grpquota**. **usrquota** enables quota controls for users, and **grpquota** works for groups. These options are usually placed in the mount entry in the **/etc/fstab** file for a particular partition. For example, to mount the **/dev/hda6** hard disk partition mounted to the **/home** directory with support for user and group quotas, you require a mount entry like the following:

```
/dev/hda6 /home ext2 defaults,usrquota,grpquota 1 1
```

You also need to create **quota.user** and **quota.group** files for each partition for which you enable quotas. These are the quota databases that hold the quota information for each user and group. You can create these files by running the **quotacheck** command with the **-a** option or the device name of the file system where you want to enable quotas. The following example creates the quota database on the hda1 hard disk partition:

```
quotacheck -a  /dev/hda1
```

edquota

You can set disk quotas using the **edquota** command. With it, you can access the quota record for a particular user and group, which is maintained in the disk quota database. You can also set default quotas that will be applied to any user or group on the file system for

edquota Option	Description
`-u`	Edits the user quota. This is the default.
`-g`	Edits the group quota.
`-p`	Duplicates the quotas of the typical user specified. This is the normal mechanism used to initialize quotas for groups of users.
`-t`	Edits the soft time limits for each file system.

TABLE 28-6 Options for **edquota**

which quotas have not been set. **edquota** will open the record in your default editor, and you can use your editor to make any changes. To open the record for a particular user, use the **-u** option and the username as an argument for **edquota** (see Table 28-6). The following example opens the disk quota record for the user **larisa**:

```
edquota -u larisa
```

The limit you set for a quota can be hard or soft. A hard limit will deny a user the ability to exceed his or her quota, whereas a soft limit will just issue a warning. For the soft limit, you can designate a grace period during which time the user has the chance to reduce their disk space below the limit. If the disk space still exceeds the limit after the grace period expires, the user can be denied access to their account. For example, a soft limit is typically 75MB, whereas the hard limit could be 100MB. Users who exceed their soft limit can have a 48-hour grace period.

The quota record begins with the hard disk device name and the blocks of memory and inodes in use. The limits segments have parameters for soft and hard limits. If these entries are 0, there are no limits in place. You can set both hard and soft limits, using the hard limit as a firm restriction. Blocks in Linux are currently about 1000 bytes. The inodes are used by files to hold information about the memory blocks making up a file. To set the time limit for a soft limit, use the **edquota** command with the **-t** option. The following example displays the quota record for **larisa**:

```
Quotas for user larisa:
/dev/hda3: blocks in use: 9000, limits (soft = 40000, hard = 60000)
 inodes in use: 321, limits (soft = 0, hard = 0)
```

quotacheck, quotaon, and quotaoff

The quota records are maintained in the quota database for that partition. Each partition that has quotas enabled has its own quota database. You can check the validity of your quota database with the **quotacheck** command. You can turn quotas on and off using the **quotaon** and **quotaoff** commands. When you start up your system, **quotacheck** is run to check the quota databases, and then **quotaon** is run to turn on quotas.

repquota and quota

As the system administrator, you can use the **repquota** command to generate a summary of disk usage for a specified file system, checking to see what users are approaching or

quota Option	Description
`-g`	Prints group quotas for the group of which the user is a member.
`-u`	Prints the user's quota.
`-v`	Displays quotas on file systems where no storage is allocated.
`-q`	Prints information on file systems where usage is over quota.

TABLE 28-7 Options for quota

exceeding quota limits. **repquota** takes as its argument the file system to check; the **-a** option checks all file systems.

```
repquota /dev/hda1
```

Individual users can use the **quota** command to check their memory use and how much disk space they have left in their quota (see Table 28-7).

Lightweight Directory Access Protocol

The Lightweight Directory Access Protocol (LDAP) is designed to implement network-accessible directories of information. In this context, the term directory is defined as a database of primarily read-only, simple, small, widely accessible, and quickly distributable information. It is not designed for transactions or updates. It is primarily used to provide information about users on a network, providing information about them such as their email address or phone number. Such directories can also be used for authentication purposes, identifying that a certain user belongs to a specified network. You can find out more information on LDAP at **ldapman.org**. You can think of an LDAP directory for users as an Internet-accessible phone book, where anyone can look you up to find your email address or other information. In fact, it may be more accurate to refer to such directories as databases. They are databases of user information, accessible over networks like the Internet. Normally, the users on a local network are spread across several different systems, and to obtain information about a user, you have to know what system the user is on and then query that system. With LDAP, user information for all users on a network is kept in the LDAP server. You only have to query the network's LDAP server to obtain information about a user. For example, Sendmail can use LDAP to look up user addresses. You can also use Firefox or Netscape to query LDAP.

NOTE *LDAP is a directory access protocol to an X.500 directory service, the OSI directory service.*

LDAP Clients and Servers

LDAP directories are implemented as clients and servers, where you use an LDAP client to access an LDAP server that manages the LDAP database. Most Linux distributions use OpenLDAP, an open-source version of LDAP (you can find out more about OpenLDAP at **openldap.org**). This package includes an LDAP server (**slapd**), an LDAP replication server (**slurpd**), an LDAP client, and tools. **slurpd** is used to update other LDAP servers on your network, should you have more than one.

LDAP Configuration Files

All LDAP configuration files are kept in the **/etc/openldap** directory. These include **slapd.conf**, the LDAP server configuration file, and **ldap.conf**, the LDAP clients and tools configuration file. To enable the LDAP server, you have to manually edit the **slapd.conf** file and change the domain value (dc) for the suffix and rootdn entries to your own network's domain address. This is the network that will be serviced by the LDAP server.

To enable LDAP clients and their tools, you have to specify the correct domain address in the **ldap.conf** file in the BASE option, along with the server's address in the HOST option (domain name or IP address). For clients, you can either edit the **ldap.conf** file directly or use the System Settings Authentication tool, clicking the Configure LDAP button on either the User Information or Authentication panel. Here you can enter the your domain name and the LDAP server's address. See the **ldap.conf** Man entry for detailed descriptions of LDAP options.

TIP *Keep in mind that the **/etc/ldap.conf** and **/etc/openldap/ldap.conf** files are not the same: **/etc/ldap.conf** is used to configure LDAP for the Nameservice Switch and PAM support, whereas **/etc/openldap/ldap.conf** is used for all LDAP clients.*

Configuring the LDAP server: /etc/slapd.conf

You configure the LDAP server with the **/etc/slapd.conf** file. Here you will find entries for loading schemas and for specifying access controls, the database directory, and passwords. The file is commented in detail, with default settings for most options, although you will have to enter settings for several. First you need to specify your domain suffix and root domain manager. The default settings are shown here:

```
suffix              "dc=my-domain,dc=com"
rootdn              "cn=Manager,dc=my-domain,dc=com"
```

In this example, the **suffix** is changed to **mytrek**, for **mytrek.com**. The **rootdn** remains the same.

```
suffix              "dc=mytrek,dc=com"
rootdn              "cn=Manager,dc=mytrek,dc=com"
```

Next you will have to specify a password with **rootpw**. There are entries for both plain text and encrypted versions. Both are commented. Remove the comment for one. In the following example the plain text password option is used, "secret":

```
rootpw              secret
# rootpw            {crypt}ijFYNcSNctBYg
```

For an encrypted password, you can first create the encrypted version with **slappasswd**. This will generate a text encryption string for the password. Then copy the generated encrypted string to the **rootpw** entry. On GNOME you can simply cut and paste from a terminal window to the **/etc/slapd.conf** file in Text Editor (Accessories). You can also redirect the encrypted string to a file and read it in later. SSHA encryption will be used by default.

```
# slappasswd
New password:
Re-enter new password:
{SSHA}0a+szaAwElK57Y8AoD5uMULSvLfCUfg5
```

The **rootpw** root password entry should then look like this:

```
rootpw      {SSHA}0a+szaAwElK57Y8AoD5uMULSvLfCUfg5
```

Use the password you entered at the **slappasswd** prompt to access your LDAP directory.

The configuration file also lists the schemas to be used. Schemas are included with the **include** directive.

```
include          /etc/openldap/schema/core.schema
include          /etc/openldap/schema/cosine.schema
include          /etc/openldap/schema/inetorgperson.schema
include          /etc/openldap/schema/nis.schema
```

NOTE *LDAP supports the Simple Authentication and Security Layer (SASL) for secure authentication with methods like MD5 and Kerberos.*

LDAP Directory Database: ldif

A record (also known as entry) in an LDAP database begins with a name, known as a *distinguishing name,* followed by a set of attributes and their values. The distinguishing name uniquely identifies the record. For example, a name could be a username and the attribute would be the user's email address, the address being the attribute's value. Allowable attributes are determined by schemas defined in the **/etc/openldap/schema** directory. This directory will hold various schema definition files, each with a **schema** extension. Some will be dependent on others, enhancing their supported classes and attributes. The basic core set of attributes is defined in the **core.schema** file. Here you will find definitions for attributes such as country name and street address. Other schemas, like **inetorgperson.schema**, specify **core.schema** as a dependent schema, making its attributes available to the classes. The inetOrgPerson schema will also define its own attributes such as jpegPhoto for a person's photograph.

Schema Attributes and Classes

Attributes and classes are defined officially by RFC specifications that are listed with each attribute and class entry in the schema files. These are standardized definitions and should not be changed. Attributes are defined by an **attributetype** definition. Each is given a unique identifying number followed by a name by which it can be referenced. Fields include the attribute description (DESC), search features such as EQUALITY and SUBSTR, and the object identifier (SYNTAX). See the OpenLDAP administrative guide for a detailed description.

```
attributetype ( 2.5.4.9 NAME ( 'street' 'streetAddress' )
      DESC 'RFC2256: street address of this object'
      EQUALITY caseIgnoreMatch
      SUBSTR caseIgnoreSubstringsMatch
      SYNTAX 1.3.6.1.4.1.1466.115.121.1.15{128} )
```

A class defines the kind of database (directory) you can create. This will specify the kinds of attributes you can include in your records. Classes can be dependent, where one class becomes and extension of another. The class most often used for LDAP databases is inetOrgPerson, defined in the **inetOrgPerson.schema** file. The term inetOrgPerson stands for Internet Organization Person, as many LDAP directories perform Internet tasks. The class is derived from the organizationalPerson class defined in **core.schema**, which includes the original attributes for commonly used fields like street address and name.

```
# inetOrgPerson
# The inetOrgPerson represents people who are associated with an
# organization in some way.  It is a structural class and is derived
# from the organizationalPerson which is defined in X.521 [X521].
objectclass ( 2.16.840.1.113730.3.2.2
    NAME 'inetOrgPerson'
      DESC 'RFC2798: Internet Organizational Person'
    SUP organizationalPerson
    STRUCTURAL
      MAY (
            audio $ businessCategory $ carLicense $ departmentNumber $
            displayName $ employeeNumber $ employeeType $ givenName $
            homePhone $ homePostalAddress $ initials $ jpegPhoto $
            labeledURI $ mail $ manager $ mobile $ o $ pager $
            photo $ roomNumber $ secretary $ uid $ userCertificate $
            x500uniqueIdentifier $ preferredLanguagei $
            userSMIMECertificate $ userPKCS12 )
      )
```

You can create your own classes, building on the standard ones already defined. You can also create your own attributes, but each attribute will require a unique object identifier (OID).

Distinguishing Names

Data in an LDAP directory is organized hierarchically, from general categories to specific data. An LDAP directory can be organized starting with countries, narrowing to states, then organizations and their subunits, and finally individuals. Commonly, LDAP directories are organized along the lines of Internet domains. In this format, the top category is the domain name extension, for instance .com or .ca. The directory then breaks down to the network (organization), units, and finally users.

This organization helps define distinguishing names that will identify the LDAP records. In a network-based organization, the top-level organization is defined by a domain component specified by the dcObject class, which includes the domainComponent (dc) attribute. Usually you define the network and extension as domain components to make up the top-level organization that becomes the distinguishing name for the database itself.

```
dc=mytrek, dc=com
```

Under the organization name is an organizational unit, such as users. These are defined as an organizationalUnitName (ou), which is part of the organizationalUnit class. The distinguishing name for the user's organizational unit would be:

```
ou=users, dc=mytrek, dc=com
```

Under the organizational unit you can then have individual users. Here the username is defined with the commonName (cn) attribute, which is used in various classes, including Person, which is part of organizationalPerson, which in turn is part of inetOrgPerson. The distinguishing name for the user **dylan** is then:

```
cn=dylan,ou=users,dc=mytrek,dc=com
```

LDIF Entries

Database entries are placed in an LDAP Interchange Format (LDIF) file. This format provides a global standard that allows a database to be accessed by any LDAP-compliant client. An LDIF file is a simple text file with an **.ldif** extension placed in the **/etc/openldap** directory. The entries for an LDIF record consist of a distinguishing name or attribute followed by a colon and its list of values. Each record begins with a distinguishing name to uniquely identify the record. Attributes then follow. You can think of the name as a record and the attributes as fields in that record. You end the record with an empty line.

Initially you create an LDIF file using any text editor, then enter the records. In the following example, the **mytrek.ldif** LDIF file contains records for users on the network.

First you create records defining your organization and organization units. These distinguishing names will be used in user-level records. You will also have to specify a manager for the database, in this case simply Manager. Be sure to include the appropriate object classes. The organization uses both the dcObject (domain component object) and organization objects. The Manager uses organizationalRole, and users use the organizationalUnit. Within each record you can have attribute definitions, like the organization attribute, **o**, in the first record, which is set to MyTrek.

```
dn: dc=mytrek,dc=com
objectclass: dcobject
objectclass: organization
dc: mytrek
o: MyTrek

dn: cn=Manager,dc=mytrek,dc=com
cn: Manager
objectclass: organizationalRole

dn: ou=users,dc=mytrek,dc=com
objectclass: organizationalUnit
ou: users
```

Individual records then follow, such as the following for **dylan**. Here the object classes are organizationalPerson and inetOrgPerson. Attributes then follow, like common name (**cn**), user ID (**uid**), organization (**o**), surname (**sn**), and street.

```
dn: cn=dylan,ou=users,dc=mytrek,dc=com
objectclass: organizationalPerson
objectclass: inetOrgPerson
cn: dylan
uid: dylan
o: MyTrek
sn: shark
street: 77777 saturn ave
```

An example of an LDIF file is shown here. The organization is mytrek.com. There are two records, one for **dylan** and the other for **chris**:

mytrek.ldif

```
dn: dc=mytrek,dc=com
objectclass: dcobject
objectclass: organization
dc: mytrek
o: MyTrek

dn: cn=Manager,dc=mytrek,dc=com
cn: Manager
objectclass: organizationalRole

dn: ou=users,dc=mytrek,dc=com
objectclass: organizationalUnit
ou: users

dn: cn=dylan,ou=users,dc=mytrek,dc=com
objectclass: organizationalPerson
objectclass: inetOrgPerson
cn: dylan
uid: dylan
o: MyTrek
sn: shark
street: 77777 saturn ave

dn: cn=chris,ou=users,dc=mytrek,dc=com
objectclass: organizationalPerson
objectclass: inetOrgPerson
cn: chris
uid: chris
o: MyTrek
sn: dolphin
street: 99999 neptune way
```

Adding the Records

Once you have created your LDIF file, you can then use the **ldapadd** command to add the records to you LDAP directory. Use the **-D** option to specify the directory to add the records to, and the **-f** option to specify the LDIF file to read from. You could use **ldapadd** to enter fields directly. The **-x** option says to use simple password access, the **-W** will prompt for the password, and the **-D** option specifies the directory manager.

```
# ldapadd -x -D "cn=Manager,dc=mytrek,dc=com" -W -f mytrek.ldif

Enter LDAP Password:

adding new entry "dc=mytrek,dc=com"

adding new entry "cn=Manager,dc=mytrek,dc=com"
```

```
adding new entry "ou=users,dc=mytrek,dc=com"

adding new entry "cn=dylan,ou=users,dc=mytrek,dc=com"

adding new entry "cn=chris,ou=users,dc=mytrek,dc=com"
```

Be sure to restart the LDAP server to have your changes take effect.

Searching LDAP

Once you have added your records, you can use the **ldapsearch** command to search your LDAP directory. The **-x** and **-W** options provide simple password access, and the **-b** option specifies the LDAP database to use. Following the options are the attributes to search for, in this case the **street** attribute.

```
# ldapsearch -x -W -D 'cn=Manager,dc=mytrek,dc=com' -b 'dc=mytrek,dc=com' street
Enter LDAP Password:
# extended LDIF
#
# LDAPv3
# base <dc=mytrek,dc=com> with scope sub
# filter: (objectclass=*)
# requesting: street

# dylan, users, mytrek.com
dn: cn=dylan,ou=users,dc=mytrek,dc=com
street: 77777 saturn ave

# chris, users, mytrek.com
dn: cn=chris,ou=users,dc=mytrek,dc=com
street: 99999 neptune way

# search result
search: 2
result: 0 Success

# numResponses: 6
# numEntries: 5
```

If you want to see all the records listed in the database, you can use the same search command without any attributes.

LDAP Tools

To actually make or change entries in the LDAP database, you use the **ldapadd** and **ldapmodify** utilities. With **ldapdelete**, you can remove entries. Once you have created an LDAP database, you can then query it, through the LDAP server, with **ldapsearch**. For the LDAP server, you can create a text file of LDAP entries using the LDAP Data Interchange Format (LDIF). Such text files can then be read in all at once to the LDAP database using the **slapadd** tool. The **slapcat** tool extracts entries from the LDAP database and saves them in an LDIF file. To reindex additions and changes, you use the **slapindex** utility. See the LDAP HOWTO at the Linux documentation project for details on using and setting up LDAP databases such as address books (**tldp.org**).

LDAP and PAM

With LDAP, you can also more carefully control the kind of information given out and to whom. Using a PAM module (**pam_ldap**), LDAP can perform user authentication tasks, providing centralized authentication for users. Login operations that users perform for different services such as POP mail server logins, system logins, and Samba logins can all be carried out through LDAP using a single PAM-secured user ID and password. To use LDAP for authentication, you need to configure PAM to use it, as well as migrate authentication files to the LDAP format. The **/usr/share/openldap/migration** directory holds scripts you can use to translate the old files into LDAP versions.

LDAP and the Name Service Switch Service

With the **libnss_ldap** module, LDAP can also be used in the Nameservice Switch (NSS) service along with NIS and system files for system database services like passwords and groups. Clients can easily enable LDAP for NSS by using the System Settings Authentication tool and selecting Enable LDAP Support in the User Information panel. You also need to make sure that the LDAP server is specified. You can also manually add **ldap** for entries in the **/etc/nsswitch.conf** file.

TIP *To better secure access to the LDAP server, you should encrypt your LDAP administrator's password. The LDAP administrator is specified in the* ***rootdn*** *entry, and its password in the* ***rootpw*** *entry. To create an encrypted password, use the* ***slappasswd*** *command. This prompts you for a password and displays its encrypted version. Copy that encrypted version in the* ***rootpw*** *entry.*

Enabling LDAP on Thunderbird

In Thunderbird, open the address book, then select File | New, and choose the LDAPD directory. Here you can enter the LDAP server. This displays a panel where you can enter the address book name, the hostname of LDAP server, the Base DN to search, and the port number, such as 389.

Pluggable Authentication Modules

Pluggable Authentication Modules (PAM) is an authentication service that lets a system determine the method of authentication to be performed for users. In a Linux system, authentication has traditionally been performed by looking up passwords. When a user logs in, the login process looks up their password in the password file. With PAM, users' requests for authentication are directed to PAM, which in turn uses a specified method to authenticate the user. This could be a simple password lookup or a request to an LDAP server, but it is PAM that provides authentication, not a direct password lookup by the user or application. In this respect, authentication becomes centralized and controlled by a specific service, PAM. The actual authentication procedures can be dynamically configured by the system administrator. Authentication is carried out by modules that can vary according to the kind of authentication needed. An administrator can add or replace

modules by simply changing the PAM configuration files. See the PAM website at **kernel.org/pub/linux/libs/pam** for more information and a listing of PAM modules. PAM modules are located in the **/lib/security** directory.

PAM Configuration Files

PAM uses different configuration files for different services that request authentication. Such configuration files are kept in the **/etc/pam.d** directory. For example, you have a configuration file for logging in to your system (**/etc/pam.d/login**), one for the graphical login (**/etc/pam.d/gdm**), and one for accessing your Samba server (**/etc/pam.d/samba**). A default PAM configuration file, called **/etc/pam.d/other**, is invoked if no services file is present. The **system-auth** file contains standard authentication modules for system services.

PAM Modules

A PAM configuration file contains a list of modules to be used for authentication. They have the following format:

```
module-type control-flag module-path module-args
```

The *module-path* is the module to be run, and *module-args* are the parameters you want passed to that module. Though there are a few generic arguments, most modules have their own. The *module-type* refers to different groups of authentication management: account, authentication, session, and password. The account management performs account verification, checking such account aspects as whether the user has access, or whether the password has expired. Authentication (**auth**) verifies who the user is, usually through a password confirmation. Password management performs authentication updates such as password changes. Session management refers to tasks performed before a service is accessed and before it is shut down. These include tasks like initiating a log of a user's activity or mounting and unmounting home directories.

TIP *As an alternative to the **/etc/pam.d** directory, you can create one configuration file called the **/etc/pam.conf** file. Entries in this file have a service field, which refers to the application that the module is used for. If the **/etc/pam.d** directory exists, **/etc/pam.conf** is automatically ignored.*

The *control-flag* field indicates how PAM is to respond if the module fails. The control can be a simple directive or a more complicated response that can specify return codes like **open_err** with actions to take. The simple directives are **requisite**, **required**, **sufficient**, and **optional**. The **requisite** directive ends the authentication process immediately if the module fails to authenticate. The **required** directive only ends the authentication after the remaining modules are run. The **sufficient** directive indicates that success of this module is enough to provide authentication unless a previous required module has failed. The **optional** directive indicates the module's success is not needed unless it is the only authentication module for its service. If you specify return codes, you can refine the conditions for authentication failure or success. Return codes can be given values such as **die** or **ok**. The **open_err** return code could be given the action **die**, which

stops all authentication and return failure. The **/etc/pam.d/vsftpd** configuration file for the FTP server is shown here:

```
#%PAM-1.0
auth required pam_listfile.so item=user sense=deny
                file=/etc/vsftpd.ftpusers onerr=succeed
auth    required  pam_stack.so service=system-auth
auth   required   pam_shells.so
account required  pam_stack.so service=system-auth
session required  pam_stack.so service=system-auth
```

NOTE *Users can provide more refined control of their files and directories by using Access Control LIsts (ACL). File systems need to be mounted with the* ***acl*** *option. The ACL tools (****acl*** *package) include* ***setfacl*** *and* ***getfacl*** *commands to set permissions. The* ***getfacl*** *command will list the user, owner, and group permissions. The* ***setfacl*** *can control access by specific users, setting read, write, and execute permissions. Use the* ***--set*** *option to add new permissions, and the* ***-m*** *option to change current ones (****-d*** *creates a default setting). Users and permissions are referenced with a colon separate list with permissions specified using the* ***r***, ***w***, *and* ***x*** *options, and* ***u***, ***g***, *and* ***o*** *referencing user, group, and other categories. The argument* ***u:chris:rw*** *would allow the user* ***chris*** *to have read and write permissions for the specified file (see* ***setfacl*** *and* ***getacl*** *Man pages for more information).*

29 CHAPTER

File Systems

Files reside on physical storage devices such as hard drives, CD-ROMs, or floppy disks. The files on each storage device are organized into a file system. The storage devices on your Linux system are presented as a collection of file systems that you can manage. When you want to add a new storage device, you need to format it as a file system and then attach it to your Linux file structure. Hard drives can be divided into separate storage devices called *partitions,* each of which has its own file system. You can perform administrative tasks on your file systems, such as backing them up, attaching or detaching them from your file structure, formatting new devices or erasing old ones, and checking a file system for problems.

To access files on a device, you attach its file system to a specified directory. This is called *mounting* the file system. For example, to access files on a floppy disk, you first mount its file system to a particular directory. With Linux, you can mount a number of different types of file systems. You can even access a Windows hard drive partition or tape drive, as well as file systems on a remote server.

Recently developed file systems for Linux now support *journaling,* which allows your system to recover from a crash or interruption easily. The **ext3**, ReiserFS, XFS, and JFS (IBM) file systems maintain a record of file and directory changes, called a *journal,* which can be used to recover files and directories in use when a system suddenly crashes due to unforeseen events such as power interruptions. Most distributions currently use the **ext3** file system as their default, though you also have the option of using ReiserFS or JFS, an independently developed journaling system.

Your Linux system is capable of handling any number of storage devices that may be connected to it. You can configure your system to access multiple hard drives, partitions on a hard drive, CD-ROM discs, DVDs, floppy disks, and even tapes. You can elect to attach these storage components manually or have them automatically mount when you boot. Automatic mounts are handled by configuring the **/etc/fstab** file. For example, the main partitions holding your Linux system programs are automatically mounted whenever you boot, whereas a floppy disk can be manually mounted when you put one in your floppy drive, though even these can also be automatically mounted. Removable storage devices like CD-ROMs, as well as removable devices like USB cameras and printers, are now handled by udev and the Hardware Abstract Layer (HAL), as described in Chapter 32 and partially discussed here.

File Systems

Although all the files in your Linux system are connected into one overall directory tree, parts of that tree may reside on different storage devices such as hard drives or CD-ROMs. Files on a particular storage device are organized into what is referred to as a *file system.* A file system is a formatted device, with its own tree of directories and files. Your Linux directory tree may encompass several file systems, each on different storage devices. On a hard drive with several partitions, you have a file system for each partition. The files themselves are organized into one seamless tree of directories, beginning from the root directory. For example, if you attach a CD-ROM to your system, a pathname will lead directly from the root directory on your hard disk partition's file system to the files in the CD-ROM file system.

TIP *With Linux you can mount file systems of different types, including those created by other operating systems, such as Windows, IBM OS, Unix, and SGI. Within Linux a variety of file systems are supported, including several journaling systems like ReiserFS and* ***ext3****.*

A file system has its files organized into its own directory tree. You can think of this as a *subtree* that must be attached to the main directory tree. The tree remains separate from your system's directory tree until you specifically connect it. For example, a floppy disk with Linux files has its own tree of directories. You need to attach this subtree to the main tree on your hard drive partition. Until they are attached, you cannot access the files on your floppy disk.

File System Hierarchy Standard (FHS)

Linux organizes its files and directories into one overall interconnected tree, beginning from the root directory and extending down to system and user directories. The organization and layout for the system directories are determined by the file system hierarchy standard (FHS). The FHS provides a standardized layout that all Linux distributions should follow in setting up their system directories. For example, there must be an **/etc** directory to hold configuration files and a **/dev** directory for device files. You can find out more about FHS, including the official documentation, at **pathname.com/fhs**. Linux distributions, developers, and administrators all follow the FHS to provide a consistent organization to the Linux file system.

Linux uses a number of specifically named directories for specialized administrative tasks. All these directories are at the very top level of your main Linux file system, the file system root directory, represented by a single slash, /. For example, the **/dev** directory holds device files, and the **/home** directory holds the user home directories and all their user files. You have access to these directories and files only as the system administrator (though users normally have read-only access). You need to log in as the root user, placing yourself in a special root user administrative directory called **/root**. From here, you can access any directory on the Linux file system, both administrative and user.

Root Directory: /

The subdirectories held in the root directory, /, are listed in Table 29-1, along with other useful subdirectories. Directories that you may commonly access as an administrator are the **/etc** directory, which holds configuration files; the **/dev** directory, which holds dynamically

Directory	Function
/	Begins the file system structure—called the root.
/boot	Holds the kernel image files and associated boot information and files.
/home	Contains users' home directories.
/sbin	Holds administration-level commands and any commands used by the root user.
/dev	Holds dynamically generated file interfaces for devices such as the terminal and the printer (see "udev: Device Files" in Chapter 31).
/etc	Holds system configuration files and any other system files.
/etc/opt	Holds system configuration files for applications in **/opt**.
/etc/X11	Holds system configuration files for the X Window System and its applications.
/bin	Holds the essential user commands and utility programs.
/lib	Holds essential shared libraries and kernel modules.
/lib/modules	Holds the kernel modules.
/media	Holds directories for mounting media-based removable file systems, such as CD-ROMs, floppy disks, USB card readers, and digital cameras.
/mnt	Holds directories for additional file systems such as hard disks.
/opt	Holds added software applications (for example, KDE on some distributions).
/proc	Process directory, a memory-resident directory that contains files used to provide information about the system.
/sys	Holds the sysfs file system for kernel objects, listing supported kernel devices and modules.
/tmp	Holds temporary files.
/usr	Holds those files and commands used by the system; this directory breaks down into several subdirectories.
/var	Holds files that vary, such as mailbox, web, and FTP files.

TABLE 29-1 Linux File System Directories

generated device files; and the **/var** directory, which holds server data files for DNS, web, mail, and FTP servers, along with system logs and scheduled tasks. For managing different versions of the kernel, you may need to access the **/boot** and **/lib/modules** directories as well as **/usr/src/linux**. The **/boot** directory holds the kernel image files for any new kernels you install, and the **/lib/modules** directory holds modules for your different kernels.

System Directories

Your Linux directory tree contains certain directories whose files are used for different system functions. For basic system administration, you should be familiar with the system program directories where applications are kept, the system configuration directory (**/etc**)

Directory	Description
/bin	Holds system-related programs.
/sbin	Holds system programs for specialized tasks.
/lib	Holds system libraries.
/etc	Holds configuration files for system and network services and applications.
/home	Holds user home directories and server data directories, such as website and FTP site files.
/media	Where removable media file systems like CD-ROMs, USB drives, and floppy disks are mounted.
/var	Holds system directories whose files continually change, such as logs, printer spool files, and lock files.
/usr	Holds user-related programs and files. Includes several key subdirectories, such as **/usr/bin**, **/usr/X11**, and **/usr/share/doc**.
/usr/bin	Holds programs for users.
/dev	Holds device files.
/sys	Holds the sysfs file system with device information for kernel-supported devices on your system.
/usr/X11	Holds X Window System configuration files.
/usr/share	Holds shared files.
/usr/share/doc	Holds documentation for applications.
/usr/share/hal	Holds configuration for HAL removable devices.
/etc/udev	Holds configuration for device files.
/tmp	Holds system temporary files.

TABLE 29-2 System Directories

where most configuration files are placed, and the system log directory (**/var/log**) that holds the system logs, recording activity on your system. Both are covered in detail in this chapter. Table 29-2 lists the system directories.

Program Directories

Directories with **bin** in the name are used to hold programs. The **/bin** directory holds basic user programs, such as login, shells (BASH, TCSH, and zsh), and file commands (`cp`, `mv`, `rm`, `ln`, and so on). The **/sbin** directory holds specialized system programs for such tasks as file system management (`fsck`, `fdisk`, `mkfs`) and system operations like shutdown and startup (`init`). The **/usr/bin** directory holds program files designed for user tasks. The **/usr/sbin** directory holds user-related system operation, such as `useradd` for adding new users. The **/lib** directory holds all the libraries your system makes use of, including the main Linux library, **libc**, and subdirectories such as **modules**, which holds all the current kernel modules.

Configuration Directories and Files

When you configure different elements of your system, such as user accounts, applications, servers, or network connections, you make use of configuration files kept in certain system directories. Configuration files are placed in the **/etc** directory.

The /usr Directory

The **/usr** directory contains a multitude of important subdirectories used to support users, providing applications, libraries, and documentation. The **/usr/bin** directory holds numerous user-accessible applications and utilities; **/usr/sbin** hold user-accessible administrative utilities. The **/usr/share** directory holds architecture-independent data that includes an extensive number of subdirectories, including those for documentation, such as **man**, **info**, and **doc** files. Table 29-3 lists the subdirectories of the **/usr** directory.

The /media Directory

The **/media** directory is used for mountpoints for removable media like CD-ROM, DVD, floppy, or Zip drives, as well as for other media-based file systems such as USB card readers, cameras, and MP3 players. These are file systems you may be changing frequently, unlike partitions on fixed disks. Most Linux systems use the Hardware Abstraction Layer (HAL) to dynamically manage the creation, mounting, and device assignment of these devices. As instructed by HAL, this tool will create floppy, CD-ROM, storage card, camera, and MP3 player subdirectories in **/media** as needed. The default subdirectory for mounting is **/media/disk**. Additional drives have an number attached to their name.

The /mnt Directory

The **/mnt** directory is usually used for mountpoints for other mounted file systems such as Windows partitions. You can create directories for any partitions you want to mount, such as **/mnt/windows** for a Windows partition.

Directory	Description
/usr/bin	Holds most user commands and utility programs.
/usr/sbin	Holds administrative applications.
/usr/lib	Holds libraries for applications, programming languages, desktops, and so on.
/usr/games	Holds games and educational programs.
/usr/include	Holds C programming language header files (**.h**).
/usr/doc	Holds Linux documentation.
/usr/local	Holds locally installed software.
/usr/share	Holds architecture-independent data such as documentation.
/usr/src	Holds source code, including the kernel source code.
/usr/X11R6	Holds X Window System–based applications and libraries.

TABLE 29-3 /usr Subdirectories

The /home Directory

The **/home** directory holds user home directories. When a user account is set up, a home directory is set up here for that account, usually with the same name as the user. As the system administrator, you can access any user's home directory, so you have control over that user's files.

The /var Directory

The **/var** directory holds subdirectories for tasks whose files change frequently, such as lock files, log files, web server files, or printer spool files. For example, the **/var** directory holds server data directories, such as **/var/www** for the Apache web server website files or **/var/ftp** for your FTP site files, as well as **/var/named** for the DNS server. The **/tmp** directory is simply a directory to hold any temporary files programs may need to perform a particular task.

The **/var** directories are designed to hold data that changes with the normal operation of the Linux system. For example, spool files for documents that you are printing are kept here. A spool file is created as a temporary printing file and is removed after printing. Other files, such as system log files, are changed constantly. Table 29-4 lists the subdirectories of the **/var** directory.

Directory	Description
/var/account	Processes accounting logs.
/var/cache	Holds application cache data for Man pages, web proxy data, fonts, or application-specific data.
/var/crash	Holds system crash dumps.
/var/games	Holds varying games data.
/var/lib	Holds state information for particular applications.
/var/local	Holds data that changes for programs installed in **/usr/local**.
/var/lock	Holds lock files that indicate when a particular program or file is in use.
/var/log	Holds log files such as **/var/log/messages** that contain all kernel and system program messages.
/var/mail	Holds user mailbox files.
/var/opt	Holds variable data for applications installed in **/opt**.
/var/run	Holds information about the system's running processes.
/var/spool	Holds applications' spool data such as that for mail, news, and printer queues, as well as `cron` and `at` jobs.
/var/tmp	Holds temporary files that should be preserved between system reboots.
/var/yp	Holds Network Information Service (NIS) data files.
/var/www	Holds web server website files.
/var/ftp	Holds FTP server FTP files.
/var/named	Holds DNS server domain configuration files.

TABLE 29-4 /var Subdirectories

The /proc File System

The **/proc** file system is a special file system that is generated in system memory. It does not exist on any disk. **/proc** contains files that provide important information about the state of your system. For example, **/proc/cpuinfo** holds information about your computer's CPU processor, **/proc/devices** lists those devices currently configured to run with your kernel, **/proc/filesystems** lists the file systems, and **/proc** files are really interfaces to the kernel, obtaining information from the kernel about your system. Table 29-5 lists the **/proc** subdirectories and files.

Like any file system, **/proc** has to be mounted. The **/etc/fstab** file will have a special entry for **/proc** with a file system type of proc and no device specified.

```
none    /proc      proc      defaults   0       0
```

TIP *You can use **sysctl**, the Kernel Tuning tool, to set proc file values you are allowed to change, like the maximum number of files, or turning on IP forwarding.*

The sysfs File System: /sys

The **sysfs** file system is a virtual file system that provides the a hierarchical map of your kernel-supported devices such as PCI devices, buses, and block devices, as well as supporting kernel modules. The **classes** subdirectory will list all your supported devices by

File	Description
/proc/*num*	There is a directory for each process labeled by its number. **/proc/1** is the directory for process 1.
/proc/cpuinfo	Contains information about the CPU, such as its type, make, model, and performance.
/proc/devices	Lists the device drivers configured for the currently running kernel.
/proc/dma	Displays the DMA channels currently used.
/proc/filesystems	Lists file systems configured into the kernel.
/proc/interrupts	Displays the interrupts in use.
/proc/ioports	Shows the I/O ports in use.
/proc/kcore	Holds an image of the physical memory of the system.
/proc/kmsg	Contains messages generated by the kernel.
/proc/loadavg	Lists the system load average.
/proc/meminfo	Displays memory usage.
/proc/modules	Lists the kernel modules currently loaded.
/proc/net	Lists status information about network protocols.
/proc/stat	Contains system operating statistics, such as page fault occurrences.
/proc/uptime	Displays the time the system has been up.
/proc/version	Displays the kernel version.

TABLE 29-5 /proc Subdirectories and Files

category, such as net and sound devices. With **sysfs** your system can easily determine the device file a particular device is associated with. This is very helpful for managing removable devices as well as dynamically configuring and managing devices as HAL and udev do. The **sysfs** file system is used by udev to dynamically generate needed device files in the **/dev** directory, as well as by HAL to manage removable device files and support as needed (HAL technically provides information only about devices, though it can use tools to dynamically change configurations as needed). The **/sys** file system type is **sysfs**. The **/sys** subdirectories organize your devices into different categories. The file system is used by **systool** to display a listing of your installed devices. The following example will list all your system devices:

```
systool
```

Like **/proc**, the **/sys** directory resides only in memory, but you still need to mount it in the **/etc/fstab** file.

```
none    /sys        sysfs     defaults   0        0
```

Device Files: /dev, udev, and HAL

To mount a file system, you have to specify its device name. The interfaces to devices that may be attached to your system are provided by special files known as *device files*. The names of these device files are the device names. Device files are located in the **/dev** directories and usually have abbreviated names ending with the number of the device. For example, **fd0** may reference the first floppy drive attached to your system. The prefix **sd** references SCSI hard drives, so **sda2** would reference the second partition on the first SCSI hard drive. In most cases, you can use the **man** command with a prefix to obtain more detailed information about this kind of device. For example, **man sd** displays the Man pages for SCSI devices. A complete listing of all device names can be found in the **devices** file located in the **linux/doc/device-list** directory at the **kernel.org** website and in the **devices.txt** file in the **/etc/usr/linux-2.4/Documentation** directory on your system. Table 29-6 lists several of the commonly used device names.

udev and HAL

Device files are no longer handled in a static way; they are now dynamically generated as needed instead. Previously a device file was created for each possible device, leading to a very large number of device files in the **/etc/dev** directory. Now, your system will detect only those devices it uses and create device files for those only, giving you a much smaller listing of device files. The tool used to detect and generate device files is udev, user devices. Each time your system is booted, udev will automatically detect your devices and generate device files for them in the **/etc/dev** directory. This means that the **/etc/dev** directory and its files are recreated each time you boot. It is a dynamic directory, no longer static. To manage these device files, you need to use udev configuration files located in the **/etc/udev** directory. This means that udev is able to also dynamically manage all removable devices; udev will generate and configure devices files for removable devices as they are attached, and then remove these files when the devices are removed. In this sense, all devices are now considered hotplugged, with fixed devices simply being hotplugged devices that are never removed.

Device Name	Description
hd	IDE hard drives; 1–4 are primary partitions; 5 and up are logical partitions
sd	SCSI hard drives
scd	SCSI CD-ROM drives
fd	Floppy disks
st	SCSI tape drives
nst	SCSI tape drives, no rewind
ht	IDE tape drives
tty	Terminals
lp	Printer ports
pty	Pseudoterminals (used for remote logins)
js	Analog joysticks
midi	Midi ports
ttyS	Serial ports
md	RAID devices
rd/c*n***d***n*	The directory that holds RAID devices is **rd**; **c***n* is the RAID controller and **d***n* is the RAID disk for that controller
cdrom	Links to your CD-ROM device file, set in **/etc/udev/rules.d**
cdrecorder	Links to your CD-R or CD-RW device file, set in **/etc/udev/rules.d**
modem	Links to your modem device file, set in **/etc/udev/rules.d**
floppy	Links to your floppy device file, set in **/etc/udev/rules.d**
tape	Links to your tape device file, set in **/etc/udev/rules.d**
scanner	Links to your scanner device file, set in **/etc/udev/rules.d**

TABLE 29-6 Device Name Prefixes

As **/etc/dev** is now dynamic, any changes you would make manually to the **/etc/dev** directory will be lost when you reboot. This includes the creation of any symbolic links such as **/dev/cdrom** that many software applications use. Instead, such symbolic links have to be configured using udev rules listed in configuration files located in the **/etc/udev/rules.d** directory. Default rules are already in place for symbolic links, but you can create rules of your own. See Chapter 32 for more details.

In addition to udev, information about removable devices like CD-ROMs and floppy disks, along with cameras and USB printers, used by applications like the desktop to dynamically interface with them, is managed by a separate utility called the Hardware Abstract Layer (HAL). HAL allows a removable device like a USB printer to be recognized no matter what particular connections it may be using. For example, you can attach a USB printer in one USB port at one time and then switch it to another later. The **fstab** file is edited using the **fstab-sync** tool, which is invoked by HAL rules in configuration files in **/usr/share/hal/fdi** directory. See Chapter 32 for more details.

HAL has a key impact on the **/etc/fstab** file used to manage file systems. No longer are entries maintained in the **/etc/fstab** file for removable devices like your CD-ROMs. These devices are managed directly by HAL using its set of storage callouts like **hal-system-storage-mount** to mount a device or **hal-system-storage-eject** to remove one. In effect you now have to use the HAL device information files to manage your removable file systems. Should you want to bypass HAL and manually configure a CD-ROM device, you simply place an entry for it in the **/etc/fstab** file.

Floppy and Hard Disk Devices

The device name for your floppy drive is **fd0**; it is located in the directory **/dev**. **/dev/fd0** references your floppy drive. Notice the numeral **0** after **fd**. If you have more than one floppy drive, additional drives are represented by **fd1**, **fd2**, and so on.

IDE hard drives use the prefix **hd**, whereas SCSI hard drives use the prefix **sd**. RAID devices, on the other hand, use the prefix **md**. The prefix for a hard disk is followed by a letter that labels the hard drive and a number for the partition. For example, **hda2** references the second partition on the first IDE hard drive, where the first hard drive is referenced with the letter **a**, as in **hda**. The device **sdb3** refers to the third partition on the second SCSI hard drive (**sdb**). RAID devices, however, are numbered from 0, like floppy drives. Device **md0** references the first RAID device, and **md1** references the second. On an IDE hard disk device, Linux supports up to four primary IDE hard disk partitions, numbered 1 through 4. You are allowed any number of logical partitions. To find the device name, you can use `df` to display your hard partitions or examine the **/etc/fstab** file.

NOTE *GNOME now manages all removable media directly with HAL, instead of using* ***fstab*** *entries.*

CD-ROM Devices

The device name for your CD-ROM drive varies depending on the type of CD-ROM you have. The device name for an IDE CD-ROM has the same prefix as an IDE hard disk partition, **hd**, and is identified by a following letter that distinguishes it from other IDE devices. For example, an IDE CD-ROM connected to your secondary IDE port may have the name **hdc**. An IDE CD-ROM connected as a slave to the secondary port may have the name **hdd**. The actual name is determined when the CD-ROM is installed, as happened when you installed your Linux system. SCSI CD-ROM drives use a different nomenclature for their device names. They begin with **scd** for SCSI drive and are followed by a distinguishing number. For example, the name of a SCSI CD-ROM could be **scd0** or **scd1**. The name of your CD-ROM was determined when you installed your system.

As noted previously, CD-ROM devices are now configured by HAL. HAL does this in a device information file in its policy configuration directory. To configure a CD-ROM device, as by adding user mount capability, you need to configure its entry in the **storage-methods.fdi** configuration file (see Chapter 31 for details). The GNOME Volume Manager uses HAL and udev to access removable media directly and Samba to provide Windows networking support. Media are mounted by **gnome-mount**, a wrapper for accessing HAL and udev, which perform the mount (**/etc/fstab** is no longer used).

Mounting File Systems

Attaching a file system on a storage device to your main directory tree is called *mounting* the device. The file system is mounted to an empty directory on the main directory tree. You can then change to that directory and access those files. If the directory does not yet exist, you have to create it. The directory in the file structure to which the new file system is attached is referred to as the *mountpoint*. For example, to access files on a CD-ROM, first you have to mount the CD-ROM.

Mounting file systems can normally be done only as the root user. This is a system administration task and should not usually be performed by a regular user. As the root user, you can, however, make a particular device, like a CD-ROM, user-mountable. In this way, any user could mount a CD-ROM. You could do the same for a floppy drive.

TIP *On GNOME, you can use the Disk Management tool on the System Settings window and menu to mount and unmount file systems, including floppy disks and CD-ROMs. On KDE, you can use the KDiskFree utility (More System Tools menu), which also lists your mountable file systems as well as their disk usage.*

Even the file systems on your hard disk partition must be explicitly mounted. When you install your Linux system and create the Linux partition on your hard drive, however, your system is automatically configured to mount your main file system whenever it starts. When your system shuts down, they are automatically unmounted. You have the option of unmounting any file system, removing it from the directory tree, and possibly replacing it with another, as is the case when you replace a CD-ROM.

Once a file system it actually mounted, an entry for it is made by the operating system in the **/etc/mstab** file. Here you will find listed all file systems currently mounted.

File System Information

The file systems on each storage device are formatted to take up a specified amount of space. For example, you may have formatted your hard drive partition to take up 3 GB. Files installed or created on that file system take up part of the space, while the remainder is available for new files and directories. To find out how much space you have free on a file system, you can use the **df** command or, on the desktop, either the GNOME System Monitor or the KDE KDiskFree utility. For the System Monitor, click the Resources tab to display a list of the free space on your file systems. KDiskFree displays a list of devices, showing how much space is free on each partition, and the percentage used.

df

The **df** command reports file system disk space usage. It lists all your file systems by their device names, how much disk space they take up, and the percentage of the disk space used, as well as where they are mounted. With the **-h** option, it displays information in a more readable format; such as measuring disk space in megabytes instead of memory blocks. The **df** command is also a safe way to obtain a listing of all your partitions, instead of using **fdisk** (because with **fdisk** you can erase partitions). **df** shows only mounted partitions, however, whereas **fdisk** shows all partitions.

```
$ df -h
Filesystem Size Used Avail Use% Mounted on
```

```
/dev/hda3   9.7G 2.8G 6.4G  31%   /
/dev/hda2   99M  6.3M 88M   7%    /boot
/dev/hda2   22G  36M  21G   1%    /home
/dev/hdc   525M  525M  0    100%  /media/disk
```

You can also use **df** to tell you to what file system a given directory belongs. Enter **df** with the directory name or **df .** for the current directory.

```
$ df .
Filesystem 1024-blocks Used Available Capacity Mounted on
/dev/hda3 297635 169499 112764 60% /
```

e2fsck and fsck

To check the consistency of the file system and repair it if it is damaged, you can use file system checking tools. **fsck** checks and repairs a Linux file system. **e2fsck** is designed to support **ext2** and **ext3** file systems, whereas the more generic **fsck** also works on any other file systems. The **ext2** and **ext3** file systems are the file systems normally used for Linux hard disk partitions and floppy disks. Linux file systems are normally **ext3**, which you use **e2fsck** to check. **fsck** and **e2fsck** take as their argument the device name of the hard disk partition that the file system uses.

```
fsck   device-name
```

Before you check a file system, be sure that the file system is unmounted. **e2fsck** should not be used on a mounted file system. To use **e2fsck**, enter **e2fsck** and the device name that references the file system. The **-p** option automatically repairs a file system without first requesting approval from the user for each repair task. The following examples check the disk in the floppy drive and the primary hard drive:

```
# e2fsck /dev/fd0
# e2fsck /dev/hda1
```

With **fsck**, the **-t** option lets you specify the type of file system to check, and the **-a** option automatically repairs systems, whereas the **-r** option first asks for confirmation. The **-A** option checks all systems in the **/etc/fstab** file.

Journaling

The **ext3** and ReiserFS file systems introduced journaling capabilities to Linux systems. Journaling provides for fast and effective recovery in case of disk crashes, instead of using **e2fsck** or **fsck**. With journaling, a log is kept of all file system actions, which are placed in a journal file. In the event of a crash, Linux only needs to read the journal file and replay it to restore the system to its previous (stable) state. Files that were in the process of writing to the disk can be restored to their original state. Journaling also avoids lengthy **fsck** checks on reboots that occur when your system suddenly loses power or freezes and has to be restarted physically. Instead of using **fsck** to manually check each file and directory, your system just reads its journal files to restore the file system.

Keeping a journal entails more work for a file system than a nonjournal method. Though all journaling systems maintain a file system's directory structure (what is known

as the *metadata*), they offer various levels of file data recovery. Maintaining file data recovery information can be time-consuming, slowing down the file system's response time. At the same time, journaling systems make more efficient use of the file system, providing a faster response time than the nonjournal **ext2** file system.

There are other kinds of journaling file systems you can use on Linux. These include ReiserFS, JFS, and XFS. ReiserFS, provides a completely reworked file system structure based on journaling (**namesys.com**). Most distributions also provide support for ReiserFS file systems. JFS is the IBM version of a journaling file system, designed for use on servers providing high throughput such as e-business enterprise servers (**http://jfs.sourceforge.net**). It is freely distributed under the GNU public license. XFS is another high-performance journaling system developed by Silicon Graphics (**oss.sgi.com/projects/xfs**). XFS is compatible with RAID and NFS file systems.

ext3 Journaling

Journaling is supported in the Linux kernel with **ext3**. The **ext3** file system is also fully compatible with the earlier **ext2** version it replaces. To create an **ext3** file system, you use the **mkfs.ext3** command. You can even upgrade **ext2** file systems to **ext3** versions automatically, with no loss of data or change in partitions. This upgrade just adds a journal file to an **ext2** file system and enables journaling on it, using the **tune2fs** command. Be sure to change the **ext2** file type to **ext3** in any corresponding **/etc/fstab** entries. The following example converts the **ext2** file system on **/dev/hda3** to an **ext3** file system by adding a journal file (**-j**).

```
tune2fs -j /dev/hda3
```

The **ext3** file system maintains full metadata recovery support (directory tree recovery), but it offers various levels of file data recovery. In effect, you are trading off less file data recovery for more speed. The **ext3** file system supports three options: **writeback**, **ordered**, and **journal**. The default is **writeback**. The **writeback** option provides only metadata recovery, no file data recovery. The **ordered** option supports limited file data recovery, and the **journal** option provides for full file data recovery. Any files in the process of being changed during a crash will be recovered. To specify a **ext3** option, use the **data** option in the **mount** command.

```
data=ordered
```

ReiserFS

Though journaling is often used to recover from disk crashes, a journal-based file system can do much more. The **ext3**, JFS, and XFS file systems only provide the logging operations used in recovery, whereas ReiserFS uses journaling techniques to completely rework file system operations. In ReiserFS, journaling is used to read and write data, abandoning the block structure used in traditional Unix and Linux systems. This gives it the capability to access a large number of small files very quickly, as well as use only the amount of disk space they need. However, efficiency is not that much better with larger files.

Mounting File Systems Automatically: /etc/fstab

File systems are mounted using the **mount** command, described in the next section. Although you can mount a file system directly with only a **mount** command, you can simplify the process by placing mount information in the **/etc/fstab** configuration file. Using entries in this file, you can have certain file systems automatically mounted whenever your system boots. For others, you can specify configuration information, such as mountpoints and access permissions, which can be automatically used whenever you mount a file system. You needn't enter this information as arguments to a **mount** command as you otherwise must. This feature is what allows mount utilities on GNOME or KDE to enable you to mount a file system simply by clicking a window icon. All the mount information is already in the **/etc/fstab** file. For example, when you add a new hard disk partition to your Linux system, you most likely want to have it automatically mounted on startup, and then unmounted when you shut down. Otherwise, you must mount and unmount the partition explicitly each time you boot up and shut down your system.

HAL and fstab

To have Linux automatically mount the file system on your new hard disk partition, you need to add only its name to the **fstab** file, except in the case of removable devices like CD-ROMs and USB printers. Removable devices are managed by HAL, using the storage policy files located in **/usr/share/hal/fdi** and **/etc/hal/fdi** directories. The devices are automatically detected by the **haldaemon** service, and are managed directly by HAL using its set of storage callouts, such as **hal-system-storage-mount** to mount a device or **hal-system-storage-eject** to remove one. In effect you now have to use the HAL device information files to manage your removable file systems. If you want different options set for the device, you should create your own **storage-methods.fdi** file in the **30user** directory. The configuration is implemented using the XML language. Check the default storage file in **10osvendors/20-storage-methods.fdi** as well as samples in **/usr/share/doc/hal***version*/**conf** directory. See Chapter 31 for examples of using HAL to set device options.

fstab Fields

An entry in an **fstab** file contains several fields, each separated from the next by a space or tab. These are described as the device, mountpoint, file system type, options, dump, and **fsck** fields, arranged in the sequence shown here:

```
<device> <mountpoint> <filesystemtype> <options> <dump> <fsck>
```

The first field is the name of the file system to be mounted. This entry can be either a device name or an **ext2** or **ext3** file system label. A device name usually begins with **/dev**, such as **/dev/hda3** for the third hard disk partition. A label is specified by assigning the label name to the tag **LABEL**, as in **LABEL=/** for an **ext2** root partition. The next field is the directory in your file structure where you want the file system on this device to be attached. The third field is the type of file system being mounted. Table 29-7 provides a list of all the different types you can mount. The type for a standard Linux hard disk partition is **ext3**. The next example shows an entry for the main Linux hard disk partition. This entry is mounted at the root directory, /, and has a file type of **ext3**:

```
/dev/hda3    /     ext3    defaults   0   1
```

Type	Description
auto	Attempts to detect the file system type automatically
minux	Minux file systems (filenames are limited to 30 characters)
ext	Earlier version of Linux file system, no longer in use
ext3	Standard Linux file system supporting long filenames and large file sizes; includes journaling
ext2	Older standard Linux file system supporting long filenames and large file sizes; does not have journaling
ntfs-3g	Windows NT, XP Vista, and 2000 file systems with write capability, NTFS-3g project.
msdos	File system for MS-DOS partitions (16-bit)
vfat	File system for Windows 95, 98, and Millennium partitions (32-bit)
reiserfs	A ReiserFS journaling file system
xfs	A Silicon Graphics (SGI) file system
ntfs	Windows NT, XP, and 2000 file systems, read only.
smbfs	Samba remote file systems, such as NFS
hpfs	File system for OS/2 high-performance partitions
nfs	NFS file system for mounting partitions from remote systems
nfs4	NFSv4 file system for mounting partitions from remote systems
umsdos	UMS-DOS file system
swap	Linux swap partition or swap file
sysv	Unix System V file systems
iso9660	File system for mounting CD-ROM
proc	Used by operating system for processes (kernel support file system)
sysfs	Used by operating system for devices (kernel support file system)
usbfs	Used by operating system for USB devices (kernel support file system)
devpts	Unix 98 Pseudo Terminals (ttys, kernel interface file system)
shmfs and **tmpfs**	Linux Virtual Memory, POSIX shared memory maintenance access (kernel interface file system)
adfs	Apple DOS file systems
affs	Amiga fast file systems
ramfs	RAM-based file systems
udf	Universal Disk Format used on CD/DVD-ROMs
ufs	Unix File System, found on Unix system (older format)

TABLE 29-7 File System Types

The following example shows a **LABEL** entry for the hard disk partition, where the label name is /:

```
LABEL=/     /       ext3    defaults    0   1
```

Auto Mounts

The file system type for a floppy may differ depending on the disk you are trying to mount. For example, you may want to read a Windows-formatted floppy disk at one time and a Linux-formatted floppy disk at another time. For this reason, the file system type specified for the floppy device is **auto**. With this option, the type of file system formatted on the floppy disk is detected automatically, and the appropriate file system type is used.

```
/dev/fd0  /media/floppy  auto   defaults,noauto   0 0
```

Mount Options

The field after the file system type lists the different options for mounting the file system. The default set of options is specified by **defaults**, and specific options are listed next to each other separated by a comma (no spaces). The **defaults** option specifies that a device is read/write (**rw**), it is asynchronous (**async**), it is a block device (**dev**), that it cannot be mounted by ordinary users (**nouser**), and that programs can be executed on it (**exec**).

Removable devices like CD-ROMs and floppy disks are now managed by HAL, the Hardware Abstraction Layer. HAL uses its own configuration files to set the options for these devices. You can place your own entries in the **/etc/fstab** file for CD-ROMs to bypass HAL. This will, however, no longer let your CD-ROMs and DVD-ROMs be automatically detected.

In a HAL configuration, a CD-ROM has **ro** and **noauto** options. **ro** specifies that the device is read-only, and **noauto** specifies it is not automatically mounted. The **noauto** option is used with both CD-ROMs and floppy drives, so they won't automatically mount, because you don't know if you have anything in them when you start up. At the same time, the HAL entries for both the CD-ROM and the floppy drive can specify where they are to be mounted when you decide to mount them. The users option allows any user to mount the system, useful for removable devices. The group option allows only users belonging to the device's group to mount it. The **fscontext** option is used by SELinux. Table 29-8 lists the options for mounting a file system. An example of a hard drive entry follows:

```
/dev/VolGroup00/LogVol00   /  ext3    defaults        1 1
```

Boot and Disk Check

The last two fields of an **fstab** entry consist of integer values. The first one is used by the **dump** command to determine if a file system needs to be dumped, backing up the file system. The second value is used by **fsck** to see if a file system should be checked at reboot, and in what order with other file systems. If the field has a value of 1, it indicates a boot partition, and 2 indicates other partitions. The 0 value means **fsck** needn't check the file system.

Option	Description
`async`	Indicates that all I/O to the file system should be done asynchronously.
`auto`	Indicates that the file system can be mounted with the `-a` option. A `mount -a` command executed when the system boots, in effect, mounts file systems automatically.
`defaults`	Uses default options: `rw`, `suid`, `dev`, `exec`, `auto`, `nouser`, and `async`.
`dev`	Interprets character or block special devices on the file system.
`group`	Users that belong to the device's group can mount it
`noauto`	Indicates that the file system can only be mounted explicitly. The `-a` option does not cause the file system to be mounted.
`exec`	Permits execution of binaries.
`owner`	Allow user that is the owner of device to mount file system.
`nouser`	Forbids an ordinary (that is, nonroot) user to mount the file system.
`fscontext`	Provide SELinux security context to those file systems without one.
`remount`	Attempts to remount an already-mounted file system. This is commonly used to change the mount flags for a file system, especially to make a read-only file system writable.
`ro`	Mounts the file system as read-only.
`rw`	Mounts the file system as read/write.
`suid`	Allows set-user-identifier or set-group-identifier bits to take effect.
`sync`	Indicates that all I/O to the file system should be done synchronously.
`user`	Enables an ordinary user to mount the file system. Ordinary users always have the following options activated: `noexec`, `nosuid`, and `nodev`.
`nodev`	Does not interpret character or block special devices on the file system.
`noexec`	Does not allow execution of binaries on the mounted file systems.
`nosuid`	Does not allow set-user-identifier or set-group-identifier bits to take effect.

TABLE 29-8 Mount Options for File Systems

fstab Sample

A copy of an **/etc/fstab** file is shown here. Notice the first line is a comment. All comment lines begin with a `#`. The entries for the **/proc** and **/sys** file systems are special entries used by your Linux operating system for managing its processes and devices; they are not actual devices. To make an entry in the **/etc/fstab** file, you can edit the **/etc/fstab** file directly. You can use the example **/etc/fstab** file shown here as a guide to show how your entries should look. The `/proc` and `swap` partition entries are particularly critical.

/etc/fstab

```
# <device> <mountpoint> <filesystemtype> <options>            <dump><fsck>
/dev/hda3       /             ext3          defaults                 0     1
none           /proc          proc          defaults                 0     0
none           /sys           sysfs         defaults                 0     0
none           /dev/pts       devpts        gid=5,mode=620           0     0
none           /dev/shm       tmpfs         defaults                 0     0
/dev/hda2      swap           swap          defaults                 0     0
/dev/hda1      /mnt/windows    vfat         defaults                 0     0
```

Partition Labels: e2label

Linux can use file system labels for **ext2** and **ext3** file systems on hard disk partitions. Thus in the **/etc/fstab** file previously shown, the first entry uses a label for its device name, as shown here. In this case, the label is the slash, /, indicating the root partition. You can change this device's label with **e2label**, but be sure to also change the **/etc/fstab** entry for it.

```
LABEL=/     /     ext3   defaults    0   1
```

For **ext2** and **ext3** partitions, you can change or add a label with the **e2label** tool or **tune2fs** with the **-L** option. Specify the device and the label name. If you change a label, be sure to change corresponding entries in the **/etc/fstab** file. Just use **e2label** with the device name to find out what the current label is. In the next example, the user changes the label of the **/dev/hda3** device to **TURTLE**:

```
e2label /dev/hda3  TURTLE
```

Windows Partitions

You can mount MS-DOS, Windows 95/98/Me, or Windows XP, NT, and 2000 partitions used by your Windows operating system onto your Linux file structure, just as you would mount any Linux file system. You have to specify the file type of **vfat** for Windows 95/98/Me and **msdos** for MS-DOS. Windows XP, NT, and 2000 use the **ntfs** file type. You may find it convenient to have your Windows partitions automatically mounted when you start up your Linux system. To do this, you need to put an entry for your Windows partitions in your **/etc/fstab** file and give it the **defaults** option or be sure to include an **auto** option. You make an entry for each Windows partition you want to mount and then specify the device name for that partition, followed by the directory in which you want to mount it. The **/mnt/windows** directory is a logical choice (be sure the **windows** directory has already been created in **/mnt**). The next example shows a standard Windows partition entry for an **/etc/fstab** file. Notice the last entry in the **/etc/fstab** file example is an entry for mounting a Windows partition.

```
/dev/hda1 /mnt/windows vfat defaults 0 0
```

For Windows XP, NT, and 2000, you specify the **ntfs** or ntfs-3g type. Be sure to have already installed NTFS support such as the Linux-NTFS Project's NTFS module (**linux-ntfs.sourceforge.net**), or the NTFS-3G project's read/write driver (**www.ntfs-3g.org**). The NTFS-3G driver provides both read and stable write support. It is included with most distributions. The **ntfs-config** configuration tool lets you set up your partitions easily on

GNOME or KDE.using NTFS-3G. The Linux-NTFS Project's kernel module is an older solution that provides only read capability.

```
/dev/hda2 /mnt/windows ntfs-3g defaults 0 0
```

NOTE *The NTFS-3G driver makes use of the Filesystem in Userspace (FUSE). FUSE implements virtual file systems in usersapce, acting as a connection to the kernel's file system management operations. With NTFS-3G, users set up a virtual file system for an NTFS partition, actions on which are handled by the kernel. FUSE has been implemented on other operating systems like Mac OSX and Windows XP for different tasks. Of note is the GmailFS file system that treats Gmail storage as if it were a file system. See* ***fuse.sourforge.net*** *for more details.*

Linux Kernel Interfaces

Your **/etc/fstab** file may also have entries for two special kernel interface file systems, devpts and tmpfs. Both provide kernel interfaces that are not supported by standard devices. The **/dev/pts** entry mounts a devpts file system for pseudoterminals. The **/dev/shm** entry mounts the tmpfs file system (also known as shmfs) to implement Linux Virtual Memory, POSIX shared memory maintenance access. This is designed to overcome the 4 GB memory limitation on current systems, extending usable memory to 64 GB.

If your **/etc/fstab** file ever becomes corrupt—say, if a line gets deleted accidentally or changed—your system will boot into a maintenance mode, giving you read-only access to your partitions. To gain read/write access so that you can fix your **/etc/fstab** file, you have to remount your main partition. The following command performs such an operation:

```
# mount -n -o remount,rw /
```

noauto

File systems listed in the **/etc/fstab** file are automatically mounted whenever you boot, unless this feature is explicitly turned off with the **`noauto`** option. Notice that the CD-ROM and floppy disks in the sample **fstab** file earlier in this chapter have a **`noauto`** option. Also, if you issue a **`mount -a`** command, all the file systems without a **`noauto`** option are mounted. If you want to make the CD-ROM user-mountable, add the **`user`** option.

```
/dev/hdc /media/cdrom iso9660 ro,noauto,user 0 0
```

TIP *The "automatic" mounting of file systems from* ***/etc/fstab*** *is actually implemented by executing a* ***`mount -a`*** *command in the system start up script (like* ***/etc/rc.d/rc.sysinit****) that is run whenever you boot. The* ***`mount -a`*** *command mounts any file system listed in your* ***/etc/fstab*** *file that does not have a* ***`noauto`*** *option. The* ***`umount -a`*** *command (which is executed when you shut down your system) unmounts the file systems in* ***/etc/fstab****.*

Mounting File Systems Manually: mount and umount

You can also mount or unmount any file system using the **`mount`** and **`umount`** commands directly (notice that **`umount`** lacks an *n*). The mount operations discussed in the previous sections use the **`mount`** command to mount a file system. Normally, file systems can be

Mount Option	Description
`-f`	Fakes the mounting of a file system. Use it to check if a file system can be mounted.
`-v`	Verbose mode. **`mount`** displays descriptions of the actions it is taking. Use with `-f` to check for any problems mounting a file system, `-fv`.
`-w`	Mounts the file system with read/write permission.
`-r`	Mounts the file system with read-only permission.
`-n`	Mounts the file system without placing an entry for it in the **mstab** file.
`-t` *type*	Specifies the type of file system to be mounted. See Table 29-7 for valid file system types.
`-a`	Mounts all file systems listed in **/etc/fstab**.
`-o` *option-list*	Mounts the file system using a list of options. This is a comma-separated list of options following `-o`. See Table 29-8 for a list of the options.

TABLE 29-9 The **`mount`** Command

mounted on hard disk partitions only by the root user, whereas CD-ROMs and floppies can be mounted by any user. Table 29-9 lists the different options for the **`mount`** command.

The mount Command

The **`mount`** command takes two arguments: the storage device through which Linux accesses the file system, and the directory in the file structure to which the new file system is attached. The *mountpoint* is the directory on your main directory tree where you want the files on the storage device attached. The *device* is a special device file that connects your system to the hardware device. The syntax for the **`mount`** command is as follows:

```
# mount device mountpoint
```

As noted previously, device files are located in the **/dev** directories and usually have abbreviated names ending with the number of the device. For example, **fd0** may refer to the first floppy drive attached to your system. The following example mounts a hard disk in the first (**hdc2**) to the **/mymedia** directory. The mountpoint directory needs to be empty. If you already have a file system mounted there, you will receive a message that another file system is already mounted there and that the directory is busy. If you mount a file system to a directory that already has files and subdirectories in it, those will be bypassed, giving you access only to the files in the mounted file system. Unmounting the file system, of course, restores access to the original directory files.

```
# mount /dev/hdc2 /mymedia
```

For any partition with an entry in the **/etc/fstab** file, you can mount the partition using only the mount directory specified in its **`fstab`** entry; you needn't enter the device filename. The **`mount`** command looks up the entry for the partition in the **fstab** file, using the directory to identify the entry and, in that way, find the device name. For example, to mount the

/dev/hda1 Windows partition in the preceding example, the **mount** command only needs to know the directory it is mounted to—in this case, **/mnt/windows**.

```
# mount /mnt/windows
```

If you are unsure about the type of file system that a disk holds, you can mount it specifying the **auto** file system type with the **-t** option. Given the **auto** file system type, **mount** attempts to detect the type of file system on the disk automatically. This is useful if you are manually mounting a floppy disk whose file system type you are unsure of (HAL also automatically detects the file system type of any removable media, including floppies).

```
# mount -t auto /dev/fd0 /media/floppy
```

The umount Command

If you want to replace one mounted file system with another, you must first explicitly unmount the one already mounted. Say you have mounted a floppy disk, and now you want to take it out and put in a new one. You must unmount that floppy disk before you can put in and mount the new one. You unmount a file system with the **umount** command. The **umount** command can take as its argument either a device name or the directory where it was mounted. Here is the syntax:

```
# umount device-or-mountpoint
```

The following example unmounts the floppy disk wherever it is mounted:

```
# umount /dev/fd0
```

Using the example where the device is mounted on the **/mydir** directory, you can use that directory to unmount the file system:

```
# umount /mydir
```

One important constraint applies to the **umount** command: you can never unmount a file system in which you are currently working. If you change to a directory within a file system that you then try to unmount, you receive an error message stating that the file system is busy. For example, suppose a CD-ROM is mounted on the **/media/disk** directory, and then you change to that **/media/disk** directory. If you decide to change CD-ROMs, you first have to unmount the current one with the **umount** command. This will fail because you are currently in the directory in which it is mounted. You have to leave that directory before you can unmount the CD-ROM.

```
# mount /dev/hdc /media/disk
# cd /media/disk
# umount /media/disk
umount: /dev/hdc: device is busy
# cd /root
# umount /media/disk
```

TIP *If other users are using a file system you are trying to unmount, you can use the* ***lsof*** *or* ***fuser*** *command to find out who they are.*

Mounting Floppy Disks

As noted previously, to access a file on a floppy disk, the disk first has to be mounted on your Linux system. The device name for your floppy drive is **fd0**, and it is located in the directory **/dev**. Entering **/dev/fd0** references your floppy drive. Notice the number **0** after **fd**. If you have more than one floppy drive, the additional drives are represented by **fd1**, **fd2**, and so on. You can mount to any directory you want. Some distributions create a convenient directory to use for floppy disks, **/media/floppy**. The following example mounts the floppy disk in your floppy drive to the **/media/floppy** directory:

```
# mount /dev/fd0 /media/floppy
```

TIP *On GNOME, you can mount a floppy drive by right-clicking the desktop background to display the desktop menu and then selecting Floppy in the Disk entry. To unmount, right-click the Floppy icon and select Eject from the pop-up menu.*

Remember, you are mounting a particular floppy disk, not the floppy drive. You cannot simply remove the floppy disk and put in another one. The **mount** command has attached those files to your main directory tree, and your system expects to find those files on a floppy disk in your floppy drive. If you take out the disk and put another one in, you get an error message when you try to access it.

To change disks, you must first unmount the floppy disk already in your disk drive. Then, after putting in the new disk, you must explicitly mount that new disk. To do this, use the **umount** command.

```
# umount /dev/fd0
```

For the **umount** or **mount** operations, you can specify either the directory it is mounted on or the **/dev/fd0** device.

```
# umount /media/floppy
```

You can now remove the floppy disk, put in the new one, and then mount it:

```
# mount /media/floppy
```

When you shut down your system, any disk you have mounted is automatically unmounted. You do not have to unmount it explicitly.

Mounting CD-ROMs

Remember, when you mount a CD-ROM or floppy disk, you cannot then simply remove it to put another one in the drive. You first have to unmount it, detaching the file system from the overall directory tree. In fact, the CD-ROM drive remains locked until you unmount it. Once you unmount a CD-ROM, you can then take it out and put in another one, which you then must mount before you can access it. When changing several CD-ROMs or floppy disks, you are continually mounting and unmounting them. For a CD-ROM, instead of using the **umount** command, you can use the **eject** command with the device name or mountpoint, which will unmount and then eject the CD-ROM from the drive.

To mount a CD-ROM, all you have to do is insert it into the drive. HAL will detect it and mount it automatically in the **/media/disk** directory.

If instead, you want to manually mount the drive from the command line with the **mount** command, you will have to first decide on a directory to mount it to (create it if it does not exist). The **/media/disk** directory is created dynamically when a disk is inserted and deleted when the disk is removed. To manually mount a disk, use the **mount** command, the device name, like **/dev/cdrom**, and the directory it is mounted to.

```
# mount /dev/cdrom  /media/cdrom1
```

If you want to manually unmount the drive, say from the command line, you can use the **umount** command and the name of the directory it is mounted on.

```
# umount /media/cdrom1
```

Or if mounted by HAL, you could use

```
# umount /media/disk
```

TIP *On GNOME, CD-ROMs are automatically mounted, though you can manually mount them by right-clicking the desktop background to display the desktop menu and then selecting CD-ROM in the Disk entry. To unmount, right-click the CD-ROM icon and select Eject from the pop-up menu.*

When you burn a CD, you may need to create a CD image file. You can access such an image file from your hard drive, mounting it as if it were another file system (even ripped images could be mounted in this way). For this, you use the **loop** option, specifying an open loop device such as **/dev/loop0**. If no loop device is indicated, **mount** will try to find an open one. The file system type is **iso9660**, a CD-ROM ISO image file type.

```
# mount -t iso9660 -o loop=/dev/loop0 image-file mount-directory
```

To mount the image file **mymusic.cdimage** to the **/mnt/mystuff** directory and make it read-only, you would use

```
# mount -t iso9660 -o ro,loop=/dev/loop0 mymusic.cdimage /mnt/mystuff
```

Once it is mounted, you can access files on the CD-ROM as you would in any directory.

TIP *You use **mkisofs** to create a CD-ROM image made up from your files or another CD-ROM.*

Mounting Hard Drive Partitions: Linux and Windows

You can mount either Linux or Windows hard drive partitions with the **mount** command. However, it is much more practical to have them mounted automatically using the **/etc/fstab** file as described previously. The Linux hard disk partitions you created during installation are already automatically mounted for you. As noted previously, to mount a Linux hard disk partition, enter the **mount** command with the device name of the partition and the directory to which you want to mount it. IDE hard drives use the prefix **hd**, and

SCSI hard drives use the prefix **sd**. The next example mounts the Linux hard disk partition on **/dev/hda4** to the directory **/mnt/mydata**.

```
# mount -t ext3 /dev/hda4 /mnt/mydata
```

You can also mount a Windows partition and directly access the files on it. As with a Linux partition, you use the **mount** command, but you also have to specify the file system type as Windows. For that, use the **-t** option, and then type **vfat** for Windows 95/98/Me (**msdos** for MS-DOS). For Windows XP, 2000, and NT, you use **ntfs** (full read-only access with limited write access). In the next example, the user mounts the Windows hard disk partition **/dev/hda1** to the Linux file structure at directory **/mnt/windows**. The **/mnt/windows** directory is a common designation for Windows file systems, though you can mount it in any directory (such as **/mnt/dos** for MS-DOS). If you have several Windows partitions, you could create a Windows directory and then a subdirectory for each drive using the drive's label or letter, such as **/mnt/windows/a** or **/mnt/windows/mystuff**. Be sure you have already created the directory before mounting the file system.

```
# mount -t vfat /dev/hda1 /mnt/windows
```

NOTE *Improved I/O support using SG_IO eliminates the need to have IDE CD-R and DVD-R drives emulate SCSI devices. Support is implemented directly by the kernel. To check that your IDE CD/DVD drives are being recognized, run* ***cdrecord*** *with the* ***-scanbus*** *option.*

Creating File Systems: mkfs, mke2fs, mkswap, parted, and fdisk

Linux provides a variety of tools for creating and managing file systems, letting you add new hard disk partitions, create CD images, and format floppies. To use a new hard drive, you will have to first partition it and then create a file system on it. You can use either **parted** or **fdisk** to partition your hard drive. To create the file system on the partitions, you use the **mkfs** command, which is a front end for various file system builders. For swap partitions, you use a special tool, **mkswap**, and to create file systems on a CD-ROM, you use the **mkisofs** tool. Linux partition and file system tools are listed in Table 29-10.

fdisk

To start **fdisk**, enter **fdisk** on the command line with the device name of the hard disk you are partitioning. This brings up an interactive program you can use to create your Linux partition. Be careful using Linux **fdisk**. It can literally erase entire hard disk partitions and all the data on those partitions if you are not careful. The following command invokes **fdisk** for creating partitions on the **hdb** hard drive.

```
fdisk   /dev/hdb
```

The partitions have different types that you need to specify. Linux **fdisk** is a line-oriented program. It has a set of one-character commands that you simply press. You may then be prompted to type in certain information and press ENTER. If you run into trouble during the **fdisk** procedure, you can press Q at any time, and you will return to the previous screen without any changes having been made. No changes are actually made to

Tool	Description
`fdisk`	Menu-driven program that creates and deletes partitions.
`cfdisk`	Screen-based interface for `fdisk`.
`parted`	Manages GNU partition, use GParted or QTParted for GUI interface.
`mkfs`	Creates a file system on a partition or floppy disk using the specified file system type. Front end to formatting utilities.
`mke2fs`	Creates an **ext2** file system on a Linux partition; use the `-j` option to create an **ext3** file system.
`mkfs.ext3`	Creates an **ext3** file system on a Linux partition.
`mkfs.ext2`	Creates an **ext2** file system on a Linux partition.
`mkfs.reiserfs`	Creates a ReiserFS journaling file system on a Linux partition (links to `mkreiserfs`).
`mkfs.jfs`	Creates a JFS journaling file system on a Linux partition.
`mkfs.xfs`	Creates a XFS journaling file system on a Linux partition.
`mkfs.dos`	Creates a DOS file system on a given partition.
`mkfs.vfat`	Creates a Windows 16-bit file system on a given partition (Windows 95/98/Me).
`mkfs.cramfs`	Creates a CRAMFS compressed flash memory file system, read-only (used for embedded devices).
`mkswap`	Sets up a Linux swap area on a device or in a file.
`mkdosfs`	Creates an MS-DOS file system under Linux.
`mkisofs`	Creates an ISO CD-ROM disk image.
`dumpe2fs`	Displays lower-level block information for a file system.
`gfloppy`	GNOME tool that formats a floppy disk (Floppy Formatter entry on the System Tools menu).
`resize2fs`	Extends the size of a partition, using unused space currently available on a disk.
`hdparm`	IDE hard disk tuner, sets IDE hard disk features.
`tune2fs`	Tunes a file system, setting features such as the label, journaling, and reserved block space.

TABLE 29-10 Linux Partition and File System Creation Tools

your hard disk until you press W. This should be your very last command; it makes the actual changes to your hard disk and then quits `fdisk`, returning you to the installation program. Table 29-11 lists the commonly used `fdisk` commands. Perform the steps in this paragraph to create a Linux partition. Press N to define a new partition; you will be asked if it is a primary partition. Press P to indicate that it is a primary partition. Linux supports up to four primary partitions. Enter the partition number for the partition you are creating and

Command	Action
a	Toggle a bootable flag
l	List known partition types
m	List commands
n	Add a new partition
p	Print the partition table
q	Quit without saving changes
t	Change a partition's system ID
w	Write table to disk and exit

TABLE 29-11 Commonly Used **fdisk** Commands

enter the beginning cylinder for the partition (this is the first number in parentheses at the end of the prompt). You are then prompted to enter the last cylinder number. You can either enter the last cylinder you want for this partition or enter a size. For example, you can enter the size as **+1000M** for 1 GB, preceding the amount with a + sign. Bear in mind that the size cannot exceed your free space. You then specify the partition type. The default type for a Linux partition is 83. If you are creating a different type of partition, such as a swap partition, press T to indicate that this is the type you want. Enter the partition number, such as 82 for a swap partition. When you are finished, press W to write out the changes to the hard disk, and then press ENTER to continue.

parted

As an alternative to **fdisk**, you can use **parted** (**gnu.org/software/parted**), which lets you manage hard disk partitions, create new ones, and delete old ones. Unlike **fdisk**, it also lets you resize partitions. To use **parted** on the partitions in a given hard drive, none of the partitions on that drive can be in use. This means that if you wish to use **parted** on partitions located on that same hard drive as your kernel, you have to boot your system in rescue mode and choose not to mount your system files. For any other hard drives, you only need to unmount their partitions and turn your swap space off with the **swapoff** command. You can then start **parted** with the **parted** command and the device name of the hard disk you want to work on. Alternatively you can use GParted on GNOME or QTParted on KDE. The following example starts **parted** for the hard disk **/dev/hda**.

```
parted  /dev/hda
```

Use the **print** command to list all your partitions. The partition number for each partition will be listed in the first column under the Minor heading. The Start and End columns list the beginning and end positions that the partition uses on the hard drive. The numbers are in megabytes, starting from the first megabyte to the total available. To create a new partition, use the **mkpart** command with either **primary** or **extended**, the file system type, and the beginning and end positions. You can create up to three primary partitions and one extended partition (or four primary partitions if there is no extended partition). The extended partition

can, in turn, have several logical partitions. Once you have created the partition, you can later use **mkfs** to format it with a file system. To remove a partition, use the **rm** command and the partition number. To resize a partition, use the **resize** command with the partition number and the beginning and end positions. You can even move a partition using the **move** command. The **help** command lists all commands.

mkfs

Once you create your partition, you have to create a file system on it. To do this, use the **mkfs** command to build the Linux file system and pass the name of the hard disk partition as a parameter. You must specify its full pathname with the **mkfs** command. Table 29-12 lists the options for the **mkfs** command. For example, the second partition on the first hard drive has the device name **/dev/hdb1**. You can now mount your new hard disk partition, attaching it to your file structure. The next example formats that partition:

```
# mkfs -t ext3 /dev/hdb1
```

The **mkfs** command is really just a front end for several different file system builders. A *file system builder* performs the actual task of creating a file system. Linux supports various file system builders, including several journaling file systems and Windows file systems. The name of a file system builder has the prefix **mkfs** and a suffix for the name of the type of file system. For example, the file system builder for the **ext3** file system is **mkfs.ext3**. For ReiserFS file systems, it is **mkfs.reiserfs** (link to **mkreiserfs** which is part of reiser-utils package). For Windows 16-bit file systems (95/98/Me), it is **mkfs.vfat**. Some of these file builders are just other names for traditional file system creation tools. For example, the **mkfs.ext2** file builder it just another name for the **mke2fs ext2** file system creation tool, and **mkfs.msdos** is the **mkdosfs** command. As **ext3** is an extension of **ext2**, **mkfs.ext3** simply invokes **mke2fs**, the tool for creating **ext2** and **ext3** file systems, and directs it to create an **ext3** file system (using the **-j** option). Any of the file builders can be used directly

Option	Description
Blocks	Specifies number of blocks for the file system. There are 1440 blocks for a 1.44 MB floppy disk.
-t *file-system-type*	Specifies the type of file system to format. The default is the standard Linux file system type, **ext3**.
file-system-options	Options for the type of file system specified. Listed before the device name, but after the file system type.
-V	Verbose mode. Displays description of each action **mkfs** takes.
-v	Instructs the file system builder program that **mkfs** invokes to show actions it takes.
-c	Checks a partition for bad blocks before formatting it (may take some time).
-l *filename*	Reads a list of bad blocks.

TABLE 29-12 The **mkfs** Options

to create a file system of that type. Options are listed before the device name. The next example is equivalent to the preceding one, creating an **ext3** file system on the **hdb1** device:

```
mkfs.ext3 /dev/hdb1
```

The syntax for the **mkfs** command is as follows. You can add options for a particular file system after the type and before the device. The block size is used for file builders that do not detect the disk size.

```
mkfs options [-t type] file-sysoptions device size
```

TIP *Once you have formatted your disk, you can label it with the* **e2label** *command as described earlier in the chapter.*

The same procedure works for floppy disks. In this case, the **mkfs** command takes as its argument the device name. It uses the **ext2** file system (the default for **mkfs**), because a floppy is too small to support a journaling file system.

```
# mkfs /dev/fd0
```

TIP *GParted provides a GUI interface for parted on GNOME. On KDE you can use QTParted. Both provide a very easy to use interface for managing partitions.*

---

mkswap

If you want to create a swap partition, you first use **fdisk** or **parted** to create the partition, if it does not already exist, and then you use the **mkswap** command to format it as a swap partition. **mkswap** formats the entire partition unless otherwise instructed. It takes as its argument the device name for the swap partition.

```
mkswap /dev/hdb2
```

You then need to create an entry for it in the **/etc/fstab** file so that it will be automatically mounted when your system boots.

CD-ROM and DVD-ROM Recording

Recording data to DVD/CD-ROM discs on Linux is now handled directly by the GNOME and KDE desktops. Simple drag and drop operations on a blank DVD/CD disc let you burn the disc. You can also use GNOME and KDE CD recording applications such as KOnCD and GNOME Toaster to create your DVD/CDs easily. All are front ends to the **mkisofs** and **cdrecord** tools. To record DVDs on DVD writers, you can use **cdrecord** for DVD-R/RW drives and the DVD+RW tools for DVD+RW/R drives. If you want to record CD-ROMs on a DVD writer, you can just use the **cdrecord**. application with many options.

The **cdrecord** application currently works only on DVD-R/RW drives; it is part of the dvdrtools package. If you want to use DVD+RW/R drives, use the DVD+RW tools such as **growisofs** and **dvd+rw-format**. Some DVD+RW tools are included in the dvd+rw-tools

package. Check the DVD+RW tools website for more information, **http://fy.chalmers.se/~appro/linux/DVD+RW**.

With the **mkisofs** command, you can create a CD image file, which you can then write to a CD-R/RW write device. Once you create your CD image file, you can write it to a CD-write device, using the **cdrecord** or **cdwrite** application. The **cdrecord** application is a more powerful.

mkisofs

To create a DVD/CD image, you first select the files you want on your CD. Then you can use **mkisofs** to create an ISO DVD?CD image of them.

mkisofs Options

You may need to include several important options with **mkisofs** to create a data CD properly. The **-o** option specifies the name of the CD image file. This can be any name you want to give it. The **-R** option specifies RockRidge CD protocols, and the **-J** option provides for long Windows 95/98/Me or XP names. The **-r** option, in addition to the RockRidge protocols (**-R**), sets standard global permissions for your files, such as read access for all users and no write access because the CD-ROM is read-only. The **-T** option creates translation tables for filenames for use on systems that are not RockRidge compliant. The **-U** option provides for relaxed filenames that are not standard ISO compliant, such as long filenames, those with more than one period in their name, those that begin with a period such as shell configuration files, and those that use lowercase characters (there are also separate options for each of these features if you just want to use a few of them). Most RPM and source code package names fall in this category. The **-iso-level** option lets you remove ISO restrictions such as the length of a filename. The **-V** option sets the volume label (name) for the CD. Finally, the **-v** option displays the progress of the image creation.

Disk Image Creation

The last argument is the directory that contains the files for which you want to make the CD image. For this, you can specify a directory. For example, if you are creating a CD-ROM to contain the data files in the **mydocs** directory, you specify that directory. This top directory will not be included, just the files and subdirectories in it. You can also change to that directory and then use **.** to indicate the current directory.

If you are creating a simple CD to use on Linux, you use **mkisofs** to first create the CD image. Here the verbose option will show the creation progress, and the **-V** option lets you specify the CD label. A CD image called **songs.iso** is created using the file located in the **newsong** directory:

```
mkisofs -v -V "Goodsongs" -o moresongs.iso  newsongs
```

If you also want to use the CD on a Windows system, you add the **-r** (RockRidge with standard global file access) and **-J** (Joliet) options:

```
mkisofs -v -r -J -V "Goodsongs" -o moresongs.iso  newsongs
```

You need to include certain options if you are using filenames that are not ISO compliant, such as those with more than 31 characters or that use lowercase characters. The **-U** option lets you use completely unrestricted filenames, whereas certain options like **-L** for

the unrestricted length will release specific restrictions only. The following example creates a CD image called **mydoc.iso** using the files and subdirectories located in the **mdoc** directory and labels the CD image with the name "Greatdocs":

```
mkisofs -v -r -T -J -U -V "Greatdocs" -o mydocuments.iso   mydocs
```

Mounting Disk Images

Once you have created your CD image, you can check to see if it is correct by mounting it as a file system on your Linux system. In effect, to test the CD image, you mount it to a directory and then access it as if it were simply another file system. Mounting a CD image requires the use of a loop device. Specify the loop device with the **loop** option as shown in the next example. Here the **mydoc.iso** is mounted to the **/media/cdrom** directory as a file system of type **iso9660**. Be sure to unmount it when you finish.

```
mount -t iso9660 -o ro,loop=/dev/loop0 mydocuments.iso /media/cdrom
```

Bootable CD-ROMs

If you are creating a bootable CD-ROM, you need to indicate the boot image file and the boot catalog to use. With the **-c** option, you specify the boot catalog. With the **-b** option, you specify the boot image. The *boot image* is a boot disk image, like that used to start up an installation procedure. For example, on the Fedora CD-ROM, the boot image is **isolinux/isolinux.bin**, and the boot catalog is **isolinux/boot.cat** (you can also use **images/boot.img** and **boot.cat**). Copy those files to your hard disk. The following example creates a bootable CD-ROM image using Red Hat Linux and Fedora distribution files located on the CD-ROM drive.

```
mkisofs -o fed7-0.iso -b isolinux/isolinux.bin -c isolinux/boot.cat \
  -no-emul-boot -boot-load-size 4 -boot-info-table \
  -v -r -R -T -J -V "Fed7"  /media/cdrom
```

cdrecord

Once **mkisofs** has created the CD image file, you can use Nautilus (the GNOME file manager) to directly burn an ISO image to a CD/DVD write disk. On the command line interface, you can, instead, use **cdrecord** to write it to a CD/DVD write disc. There is a command called **dvdrcecord**, but this is just a script that calls **cdrecord**, which now writes to both DVD and CD media. If you have more than one CD-writer device, you should specify the DVD/CD-R/RW drive to use by indicating its device name. In this example, the device is an IDE CD-R located at **/dev/hdc**. The **dev=** option is used to indicate this drive. The final argument for **cdrecord** is the name of the CD image file.

```
cdrecord  dev=/dev/hdc  mydocuments.iso
```

In this example, a SCSI rewritable CD-RW device with the device **/dev/scd0** is used.

```
cdrecord  dev=/dev/scd0  mydocuments.iso
```

If you are creating an audio CD, use the **-audio** option, as shown here. This option uses the CD-DA audio format:

```
cdrecord  dev=/dev/hdc -audio moresongs.iso
```

TIP *The* ***dummy*** *option for* ***cdrecord*** *lets you test the CD writing operation for a given image.*

DVD+RW Tools

The primary DVD+RW tool is **growisofs**, with which you create DVD+RW/R discs. Two other minor supporting tools are also included, a formatter, `dvd+rw-format`, and a compatibility tool, `dvd+rw-booktype`. See the dvd+rw-tools page in **/usr/share/doc** for detailed instructions.

The **growisofs** tool functions like the **mkisofs** tool, except that it writes directly to the DVD+RW/R disc, rather than to an image. It has the same options as **mkisofs**, with a few exceptions, and is actually a front end to the **mkisofs** command. There is, of course, no **-o** option for specifying a disk image. You specify the DVD device instead. For example, to write the contents of the **newsongs** directory to a DVD+RW disc, you would use **growisofs** directly.

```
growisofs -v -V "Goodsongs" -Z /dev/hdc  newsongs
```

The device is specified by its name, usually **/dev/scd0** for the first SCSII device or **/dev/hdc** for the first secondary IDE drive. Recall that IDE DVD writers are configured as SCSI devices when your system boots up. **growisofs** provides a special **-Z** option for burning an initial session. For multisessions (DVD-RW), you can use the **mkisofs -M** option. If you want to reuse a DVD-RW disc, just overwrite it. You do not have to reformat it.

To burn an ISO image file to the disc, use the **-Z** option and assign the ISO image to the device.

```
growisofs -v -V "Goodsongs" -Z /dev/hdc=moresongs.iso
```

Though **growisofs** will automatically format new DVD+RW discs, the DVD+RW tools also include the `dvd+rw-format` tool for explicitly formatting new DVD+RW (read/write) discs, preparing them for writing. This is done only once, and only for DVD+RW discs that have never been used before. DVD+R discs do not need any formatting.

The `dvd+rw-booktype` tool sets the compatibility setting for older DVD-ROM readers that may not be able to read DVD+RW/R discs.

Mono and .NET Support

With Mono, Linux now provides .NET support, along with .NET applications like the Beagle desktop search tool and the F-Spot photo management tool. Mono provides an open source development environment for .NET applications. The Mono project is an open source project supported by Novell that implements a .NET framework on Unix, Linux, and OS X systems. Currently Mono 1.2 and 2.0 are included. Mono 1.2 corresponds generally with .NET 1.1 features, and Mono 2.0 corresponds with .NET 2.0. See **mono-project.com** for detailed information.

Mono is implemented on Linux using several components. These include the basic Mono .NET application, including Mono tools like the Mono certification manager (certmgr); the Global Assemblies Cache Manager tool (gacutil) for making assemblies available at runtime; and mcs, the Mono C# compiler. There are several additional tools for distinct features like

visual basic support, SQL database queries, and .NET web support. The Mono language testing tool, NUnit, is also included.

Configuration is found in the **/etc/mono/config** file, which is an XML-like file that maps DLL references to Linux libraries. The **/etc/mono** file also contains configuration files for Mono 1.0 and 2.0. Mono is installed in **/usr/lib/mono**. In the corresponding **1.0** and **2.0** directories you will find the DLL and EXE .NET support assemblies for different Mono applications. Other directories will hold .NET DLLs and configuration for different applications and services, including evolution, dbus, and gtk.

Local configuration information and runtime applications are placed in the user's **.config** directory.

30 CHAPTER

RAID and LVM

With onset of cheap, efficient, and very large hard drives, even simple home systems may employ several hard drives. The use of multiple hard drives opens up opportunities for ensuring storage reliability as well as more easily organizing access to your hard disks. Linux provides two methods for better managing your hard disks: Logical Volume Management (LVM) and Redundant Arrays of Independent Disks (RAID). LVM is a method for organizing all your hard disks into logical volumes, letting you pool the storage capabilities of several hard disks into a single logical volume. Your system then sees one large storage device, and you do not have to micromanage each underlying hard disk and its partitions. LVM is perhaps the most effective way to add hard drives to your system, creating a large accessible pool of storage. RAID is a way of storing the same data in different places on multiple hard disks. These multiple hard drives are treated as a single hard drive. They include recovery information that allows you to restore your files should one of the drives fail. The two can be mixed, implementing LVM volumes on RAID arrays. LVM provides flexibility, and RAID can provide data protection.

With LVM you no longer have to keep track of separate disks and their partitions, trying to remember where files are stored on what partitions located in what drive. Partitions and their drives are combined into logical file systems that you can attach to your system directory tree. You can have several logical file systems, each with their own drives and/or partitions.

In a system with several hard drives with both LVM and RAID you can combine the hard drives into one logical file system that accesses the storage as one large pool. Files are stored in a single directory structure, not on directories on a particular partition. Instead of mounting file systems for each individual hard drive, there is only one file system to mount for all the hard drives. LVM has the added advantage of letting you implement several logical file systems on different partitions across several hard drives.

RAID is best suited to desktops and servers that hold multiple hard drives and require data recovery. The most favored form of RAID, RAID 5, requires a minimum of three hard drives. RAID, with the exception of RAID 0, provides the best protection against hard drive failure and is considered a necessity for storage-intensive tasks such as enterprise, database, and Internet server operations. It can also provide peace of mind for smaller operations, providing recovery from hard disk failure. Keep in mind that there are different forms of RAID, each with advantages and weaknesses. RAID 0 provides no recovery capabilities at all. After setting up a RAID array, you can implement LVM volumes on the array.

In comparison to LVM, RAID can provide faster access for applications that work with very large files, such as multimedia, database, or graphics applications. But for normal operations, LVM is just as efficient as RAID. LVM, though, requires running your Linux system and configuring it from your Linux operating system. RAID, which is now supported at the hardware level on most computers, is easier to set up, especially a simple RAID 0 operation that merely combines hard drives into one drive.

Logical Volume Manager (LVM)

For easier hard disk storage management, you can set up your system to use the Logical Volume Manager (LVM), creating LVM partitions that are organized into logical volumes to which free space is automatically allocated. Logical volumes provide a more flexible and powerful way of dealing with disk storage, organizing physical partitions into logical volumes in which you can easily manage disk space. Disk storage for a logical volume is treated as one pool of memory, though the volume may in fact contain several hard disk partitions spread across different hard disks. Adding a new LVM partition merely increases the pool of storage accessible to the entire system. The original LVM package was developed for kernel 2.4. The current LVM2 package is used for kernel 2.6. Check the LVM HOWTO at **tldp.org** for detailed examples.

NOTE *Many distributions include Red Hat's* ***system-config-lvm*** *tool. With this tool you can easily manage your LVM devices with a simple GUI interface.*

LVM Structure

In an LVM structure, LVM physical partitions, also known as *extents,* are organized into logical groups, which are in turn used by logical volumes. In effect, you are dealing with three different levels of organization. At the lowest level, you have physical volumes. These are physical hard disk partitions that you create with partition creation tools such as **`parted`** or **`fdisk`**. The partition type will be a Linux LVM partition, code **8e**. These physical volumes are organized into logical groups, known as volume groups, that operate much like logical hard disks. You assign collections of physical volumes to different logical groups.

Once you have your logical groups, you can then create logical volumes. Logical volumes function much like hard disk partitions on a standard setup. For example, on the **turtle** group volume, you could create a **/var** logical volume, and on the **rabbit** logical group, you could create **/home** and **/projects** logical volumes. You can have several logical volumes on one logical group, just as you can have several partitions on one hard disk.

You treat the logical volumes as you would any ordinary hard disk partition. You create a file system on one with the **`mkfs`** command, and then you can mount the file system to use it with the **`mount`** command. The Linux file system type could be **ext3**.

Storage on logical volumes is managed using what are known as extents. A logical group defines a standard size for an extent, say 4 MB, and then divides each physical volume in its group into extents of that size. Logical volumes are, in turn, divided into extents of the same size, which are then mapped to those on the physical volumes.

Logical volumes can be linear, striped, or mirrored. The mirror option will create a mirror copy of a logical volume, providing a restore capability. The striped option lets you automatically distribute your logical volume across several partitions as you would

a RAID device. This adds better efficiency for very large files but is complicated to implement. As on a RAID device, stripe sizes have to be consistent across partitions and, as LVM partitions can be of any size, the stripe sizes have to be carefully calculated. The simplest approach is to use a linear implementation, much like a RAID 0 device, treating the storage as one large ordinary drive, with storage accessed sequentially.

There is one restriction and recommendation for logical volumes. The boot partition cannot be part of a logical volume. You still have to create a separate hard disk partition as your boot partition with the **/boot** mountpoint in which your kernel and all needed boot files are installed. In addition, it is recommended that you not place your root partition on a logical volume. Doing so can complicate any needed data recovery. This is why a default partition configuration for many distributions, set up during installation, will include a separate **/boot** partition of 100 MB of type **ext3**, whereas the root and swap partitions will be installed on logical volumes. There will be two partitions, one for the logical group (LVM physical volume, **pv**) holding both swap and root volumes, and another for the boot partition (**ext3**). The logical volumes will in turn both be **ext3** file systems.

Creating LVMs During Installation

Many distributions also allow you to create LVM during installation. LVM may even be the default configuration. First, you create physical LVM partitions, then the volume groups you place these partitions in, and then from the volume groups you create the logical volumes, for which you then specify mountpoints and file system types. You can create LVM partitions during the installation process. Create an LVM physical partition on your hard disk. Once you have created the LVM physical partition, you create your logical volumes. You first need to assign the LVM physical partitions to volume groups, which are essentially logical hard drives. Once the volume groups are created, you are ready to create your logical volumes. You can create several logical volumes within each group. The logical volumes function like partitions. You will have to specify a file system type and mountpoint for each logical volume you create, and you need at least a swap and root volume. The file system type for the swap volume is **swap**, and for the root volume it's a standard Linux file system type like **ext3**.

Distribution Configuration Tools

Many distributions have their own LVM configuration tools that provide an easy-to-use GUI interface for managing your LVM file systems. For Red Hat and Fedora you can use system-config-lvm. Tools such as these provide detailed context help as well as online tutorial and manuals. They access the underlying LVM commands described in this chapter to perform their tasks. It is usually preferable to use the distribution LVM configuration tools to manage your LVM file systems than try to use the LVM commands.

LVM Tools: Using the LVM Commands

You can use the collection of LVM tools to manage your LVM volumes, adding new LVM physical partitions and removing current ones (see Table 30-1). The distribution GUI LVM tools, like system-config-lvm, are actually GUI interfaces for the LVM tools. You can either use LVM tools directly or use the `lvm` command to generate an interactive shell from which you can run LVM commands. There are Man pages for all the LVM commands. LVM maintains configuration information in the **/etc/lvm/lvm.conf** file, where you can configure LVM options such as the log file, the configuration backup directory, or the directory for LVM devices (see the **lvm.conf** Man page for more details).

Command	Description
`lvm`	Open an interactive shell for executing LVM commands
`lvmdiskscan`	Scan all disks for LVM physical partitions
`lvdisplay`	Display detailed information about logical volumes
`lvcreate`	Create logical volumes
`lvrename`	Rename a logical volume
`lvchange`	Modify a logical volume
`lvextend`	Extend the size of a logical volume
`lvreduce`	Reduce the size of a logical volume
`lvremove`	Remove logical volumes
`lvs`	List logical volumes with detailed information
`lvresize`	Change the size of a logical volume
`lvscan`	Scan system for logical volumes
`pvdisplay`	Display detailed information about LVM physical partition
`pvchange`	Modify an LVM physical partition
`pvcreate`	Create LVM physical partitions
`pvmove`	Move contnest of an LVM physical partition to another partition
`pvremove`	Delete LVM physical partitions
`pvs`	List physical partitions with detailed information
`pvresize`	Resize a physical partition
`pvscan`	Scan system for physical partitions
`vgdisplay`	Display detailed information about volume groups
`vgexport`	Activate a volume group
`vgimport`	Make an exported volume group known to a new system. Useful for moving an activated volume group from one system to another.
`vgmerge`	Combine volume groups
`vgreduce`	Remove physical partitions from a volume group
`vgremove`	Delete a volume group
`vgs`	List volume groups with detailed information
`vgslit`	Split a volume group
`vgscan`	Scan system for volume groups
`vgck`	Check volume groups
`vgrename`	Rename a volume group
`vgcfgbackup`	Backup volume group configuration (metadata)
`vgcfgrestore`	Restore volume group configuration (metadata)

Table 30-1 LVM Commands

Displaying LVM Information

You can use the **pvdisplay**, **vgdisplay**, and **lvdisplay** commands to show detailed information about a physical partition, volume groups, and logical volumes. **pvscan**, **vgscan**, and **lvscan** list your physical, group, and logical volumes.

Managing LVM Physical Volumes with the LVM Commands

A physical volume can be any hard disk partition or RAID device. A RAID device is seen as a single physical volume. You can create physical volumes either from a single hard disk or from partitions on a hard disk. On very large systems with many hard disks, you would more likely use an entire hard disk for each physical volume.

To initialize a physical volume on an entire hard disk, you use the hard disk device name, as shown here:

```
pvcreate /dev/sdc
```

If you are using a single partition for an entire drive, you create a new physical volume using the partition's device name, as shown here:

```
pvcreate /dev/sdc1
```

This will create one physical partition, pv, called **sdc1** on the **sdc** hard drive (the third Serial ATA drive, drive c).

To initialize several drives, just list them. The following creates two physical partitions, sdc1 and sdd1.

```
pvcreate /dev/sdc1 /dev/sdd1
```

You can also use several partitions on different hard drives. This is a situation in which your hard drives each hold several partitions. This condition occurs often when you are using some partitions on your hard drive for different purposes like different operating systems, or if you want to distribute your logical group across several hard drives. To initialize these partitions at once, you simply list them.

```
pvcreate /dev/hda3 /dev/hdb1 /dev/hdb2
```

Once you have initialized your partitions, you have to create LVM groups on them.

Managing LVM Groups

Physical LVM partitions are used to make up a volume group. You can manually create a volume group using the **vgcreate** command and the name of the group along with a list of physical partitions you want in the group.

If you are then creating a new volume group to place these partitions in, you can include them in the group when you create the volume group with the **vgcreate** command. The volume group can use one or more physical partitions. The default install configuration described previously used only one physical partition for the **VolGroup00**. In the following example, a volume group called **mymedia** that is made up two physical volumes, **sdb1** and **sdc1**.

```
vgcreate mymedia  /dev/sdb1 /dev/sdc1
```

The previous example sets up a logical group on two serial ATA hard drives, each with its own single partition. Alternatively, you can set up a volume group to span partitions on several hard drives. If you are using partitions for different functions, this approach gives you the flexibility for using all the space available across multiple hard drives. The following example creates a group called **rabbit** consisting of three physical partitions, **/dev/hda3**, **/dev/hdb4**, and **/dev/hdb4**:

```
vgcreate rabbit  /dev/hda3 /dev/hdb2 /dev/hdb4
```

If you to later want to add a physical volume to a volume group, you use the **vgextend** command. The **vgextend** command adds a new partition to a logical group. In the following example, the partition **/dev/hda**3 is added to the volume group **rabbit**. In effect, you are extending the size of the logical group by adding a new physical partition.

```
vgextend  rabbit  /dev/hda3
```

To add an entire new drive to a volume group, you follow a similar procedure. The following example adds a fifth serial ATA hard drive, **sde**, first creating a physical volume on it and then adding that volume, **sde1**, to the **mymedia** volume group.

```
pvcreate /dev/sde1
vgextend mymedia /dev/sde1
```

To remove a physical partition, first remove it from its logical group. You may have to use the **pmove** command to move any data off the physical partition. Then use the **vgreduce** command to remove it from its logical group.

You can, in turn, remove a entire volume group by first deactivating it with **vgchange -a n** and then using the **vgremove** command.

Activating Volume Groups

Whereas in a standard file system structure, you mount and unmount hard disk partitions, with an LVM structure, you activate and deactivate entire volume groups. The group volumes are inaccessible until you activate them with the **vgchange** command with the **-a** option. To activate a group, first reboot your system, and then enter the **vgchange** command with the **-a** option and the **y** argument to activate the logical group (an **n** argument will deactivate the group).

```
vgchange -a  y  rabbit
```

Managing LVM Logical Volumes

To create logical volumes, you use the **lvcreate** command and then format your logical volume using the standard formatting command like **mkfs.ext3**.

With the **-n** option you specify the volume's name, which functions like a hard disk partition's label. You use the **-L** option to specify the size of the volume. There are other options for implementing features such as whether to implement a linear, striped, or mirrored volume, or to specify the size of the extents to use. Usually the defaults work well. The following example creates a logical volume named **projects** on the **rabbit** logical group with a size of 20 GB.

```
lvcreate -n projects  -L 20GB rabbit
```

The following example sets up a logical volume on the **mymedia** volume group that is 540 GB in size. The **mymedia** volume group is made up of two physical volumes, each on 320 GB hard drives. In effect the two hard drives are logically seen as one.

```
lvcreate -n myvideos  -L 540GB mymedia
```

Once you have created your logical volume, you then have to create a file system to use on it. The following creates an **ext3** file system on the **myvideos** logical volume.

```
mkfs.ext3 myvideos
```

You can remove a logical volume with the **`lvremove`** command. With **`lvextend`**, you can increase the size of the logical volume, and **`lvreduce`** will reduce its size.

LVM Device Names: /dev/mapper

The **device-mapper** driver is used by LVM to set up tables for mapping logical devices to hard disk. The device name for a logical volume is kept in the **/dev/mapper** directory and has the format *logical group –logical volume*. In addition, there will be a corresponding device folder for the logical group, which will contain logical volume names. These device names are links to the **/dev/mapper** names. For example, the **mypics** logical volume in the **mymedia** logical group has the device name, **/dev/mapper/mymedia-mypics**. There will be a corresponding folder called **/dev/mymedia**. The device name **/dev/mymedia/mypics** is a link to the **/dev/mapper/mymedia-mypics** device name. You can just as easily use the link as shown in this chapter, as the original device name. The snapshot device described later in this chapter would have the device name **/dev/mapper/mymedia-mypicssnap1** with the link device name being **/dev/mymedia/mypicssnap**.

NOTE *You can back up volume group meta data (configuration) using the **vgcfgbackup** command. This does not backup your logical volumes (no content). Meta data backups are stored in **/etc/lvm/backup**, and can be restored using **vgcfgrestore**.*

LVM Example for Multiple Hard Drives

With hard drives becoming cheaper and the demand for storage increasing, many systems now use multiple hard drives. To manage multiple hard drives, partitions on each used to have to be individually managed, unless you implemented a RAID system. RAID allows you to treat several hard disks as one storage device, but there are restrictions on the size and kinds of devices you can combine. Without RAID, each hard drive had to be managed separately, with files having to fit into remaining storage as the drives filled up.

With LVM, you no longer have such restrictions. You can combine hard disks into a single storage device. This method is also flexible, letting you replace disks without losing any data, as well adding new disks to automatically increase your storage (or replace smaller disks with larger ones).

For example, say you want to add two hard disks to your system, but you want to treat the storage in both logically instead of having to manage partitions in each. LVM lets you treat the combined storage of both hard drives as one giant pool. In effect, two 500 GB drives can be treated as one 1 terabyte storage device.

In this example, the Linux system makes use of three hard drives. The Linux system and boot partitions are on the first hard drive, **sda**. Added to this system are two hard drives, **sdb and sdc**, which will make up an LVM storage device to be added to the system.

The steps involved in creating and accessing logical volumes are described in following commands. In this example there are two hard disk drives that will be combined into one LVM drive. The hard drives are Serial ATA drives identified on the systems as **sdb** and **sdc**. Each drive is first partitioned with a single LVM physical partition. Use a partition creation tool like **fdisk** or **parted** to create the physical partitions on the hard disks **sdb** and **sdc**. In this example, you create the partitions **sdb1** and **sdc1**.

You first initialize the physical volumes with the **pvcreate** command. The **sda1** and **sda2** partitions in the **sda** entry are reserved for the boot and root partitions and are never initialized.

```
pvcreate /dev/sda1 /dev/sdb1 /dev/sdc1
```

You then create the logical groups you want using the **vgcreate** command. In this case there is one logical group, **mymedia**. The **mymedia** group uses **sdb1** and **sdc1**. If you create a physical volume later and want to add it to a volume group, you use the **vgextend** command.

```
vgcreate mymedia  /dev/sdb1 /dev/sdc1
```

You can now create the logical volumes in each volume group, using the **lvcreate** command. In this example two logical volumes are created, one for **myvideos** and another for **mypics**. The corresponding **lvcreate** commands are shown here:

```
lvcreate  -n myvideo   -l 540GB    mymedia
lvcreate  -n mypics    -l 60GB    mymedia
```

Then you can activate the logical volumes. Reboot and use **vgchange** with the **-a y** option to activate the logical volumes.

```
vgchange -a y mymedia
```

You can now make file systems for each logical volume.

```
mkfs.ext3 myvideo
mkfs.ext3 mypics
```

Then you can mount the logical volumes. In this example they are mounted to subdirectories of the same name in **/mydata**.

```
mount –t ext3 /dev/mymedia/mypics  /mydata/mypics
mount –t ext3 /dev/mymedia/myvideo /mydata/myvideo
```

Using LVM to Replace Drives

LVM can be very useful when you need to replace an older hard drive with a new one. Hard drives are expected to last about six years on the average. You might also want to just replace the older drive with a larger one (available hard drive storage sizes double every year or so). To replace an on-boot hard drive is very easy. To replace a boot drive becomes much more complicated.

To replace a drive, simply incorporate the new drive into your logical volume. The size of your logical volume will increase accordingly. You can use the **pmove** command to move data from the old drive to the new one. Then, issue commands to remove the old one. From the user and system point of view there will be no changes. Files from your old drive will still be stored in the same directories, though the actual storage will be implemented on the new drive.

Replacement with LVM become more complicated if you want to replace your boot drive, the hard drive your system starts up from and that holds your Linux kernel. The boot drive contains a special boot partition and the master boot record. The boot partition cannot be part of any LVM volume. You first have to create a boot partition on the new drive using a partition tool like **parted** or **fdisk**, labeling it as boot. The boot drive is usually very small, about 200 MB. Then mount the partition on your system and copy the contents of your **/boot** directory to it. Then you add the remainder of the disk to your logical volume and logically remove the old disk, copying the contents of the old disk to the new one. You still have to boot with the Linux rescue DVD (or install DVD in rescue mode) and issue the **grub-install** command to install the master boot record on your new drive. You can then boot from the new drive.

LVM Example for Partitions on Different Hard Drives

In a more complex implementation, you can use partitions on different hard drives for the same logical volumes. For example, if you have physical volumes consisting of the hard disk partitions **hda2**, **hda3**, **hdb1**, **hdb2**, and **hdb3** on two hard disks, **hda** and **hdb**, you can assign some of them to one logical group and others to another logical group. The partitions making up the different logical groups can be from different physical hard drives. For example, **hda2** and **hdb3** could belong to the logical group **turtle** and **hda3**, **hdb2**, and **hdb3** could make up a different logical group, say **rabbit**. The logical group name could be any name you want to give it. It is much like naming a hard drive.

NOTE *The examples here use the **hd** prefix for referencing drives. Some distributions, like Fedora, have dropped the **hd** prefix and use the **sd** prefix for both ATA and Serial hard drives.*

Using the example in Figure 30-1, the steps involved in creating and accessing logical volumes are described in following commands. First, use a partition creation tool like **fdisk** or **parted** to create the physical partitions on the hard disks **hda** and **hdb**. In this example, you create the partitions **hda1**, **hda2**, **hda3**, **hdb1**, **hdb2**, **hdb3**, and **hdb4**.

Next, initialize the physical volumes with the **pvcreate** command. The **hda1** and **hda2** partitions are reserved for the boot and root partitions and are not initialized.

```
pvcreate /dev/hda3 /dev/hdb1 /dev/hdb2
pvcreate /dev/hdb3 /dev/hdb4
```

You then create the logical groups you want using the **vgcreate** command. In this case there are two logical groups, **turtle** and **rabbit**. The **turtle** group uses **hdb1** and **hdb3**, and **rabbit** uses **hda3**, **hdb2**, and **hdb4**. If you create a physical volume later and want to add it to a volume group, you use the **vgextend** command.

```
vgcreate turtle  /dev/hdb1 /dev/hdb3
vgcreate rabbit  /dev/hda3  /dev/hdb2 /dev/hdb4
```

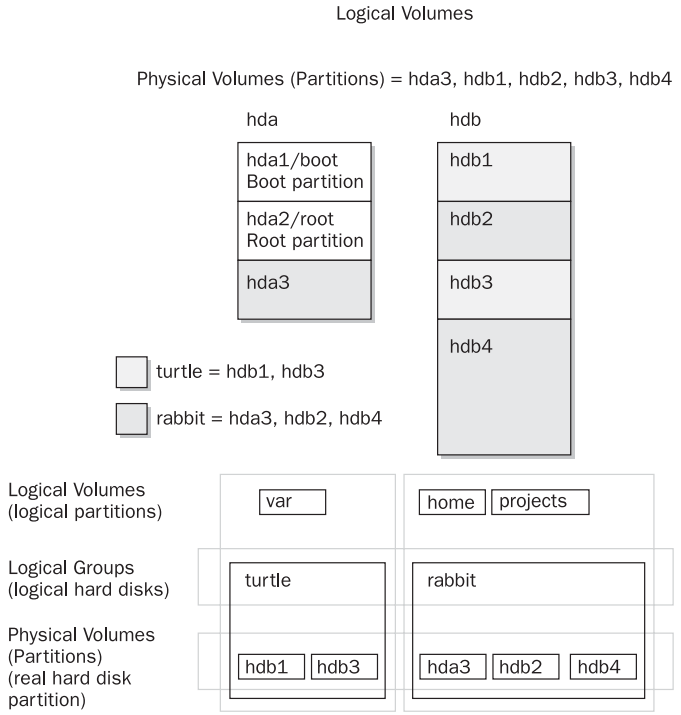


FIGURE 30-1 Logical volume using multiple partitions on different hard drives

You can now create the logical volumes in each volume group, using the **lvcreate** command.

```
lvcreate  -n var        -l 2000M     turtle
lvcreate  -n home       -l 50000M    rabbit
lvcreate  -n projects   -l 20000M    rabbit
```

Then you can activate the logical volumes. Reboot and use **vgchange** with the **-a y** option to activate the logical volumes.

```
vgchange -a y turtle rabbit
```

You can now make file systems for each logical volume.

```
mkfs.ext3 var
mkfs.ext3 home
mkfs.ext3 projects
```

Then you can mount the logical volumes.

```
mount –t ext3 /dev/turtle/var  /var
mount –t ext3 /dev/rabbit/home /home
mount –t ext3 /dev/rabbit/projects /mnt/myprojects
```

LVM Snapshots

A snapshot records the state of the logical volume at a designated time. It does not create a full copy of data on the volume, just changes since the last snapshot. A snapshot defines the state of the data at a given time. This allows you to backup the data in a consistent way. Also, should you need to restore a file to its pervious version, you can use the snapshot of it. Snapshots are treated as logical volume and can be mounted, copied, or deleted.

To create a snapshot you use the lvcreate command with the -s option. In this example the snapshot is given the name mypics-snap1 (**-n** option). You need to specify the full device name for the logical group you want to create the snapshot for. Be sure there is enough free space available in the logical group for the snapshot. In this example, the snapshot logical volume is created in the **/dev/mymedia** logical group. It could just as easily be created in any other logical group. Though a snapshot normally uses very little space, you have to guard against overflows. If the snapshot is allocated the same size as the original, it will never overflow. For systems where little of the original data changes, the snapshot can be very small. The following example allocate one third the size of the original (60 GB).

```
lvcreate  -s –n mypics-snap1 -l 20GB /dev/mymedia
```

You can then mount the snapshot as you would any other file system.

```
mount /dev/mymedia/mypicsnap1 /mysnaps
```

To delete a snapshot you use the **lvremove** command, removing it like you would any logical volume.

```
lvremove  -f /dev/mymedia/mypicsnap1
```

Snapshots are very useful for making backups, while a system is still active. You can use **tar** or **dump** to backup the mounted snapshot to a disk or tape. All the data from the original logical volume will be included, along with the changes noted by the snapshot.

Snapshots also allow you to perform effective undo operations. You can create a snapshot of a logical volume. Then unmount the original and mount the snapshot in its place. Any changes you make will be performed on the snapshot, not the original. Should problems occur, unmount the snapshot and then mount the original. This restores the original state of your data. You could also do this using several snapshots, restoring to a previous snapshot. With this procedure, you could test new software on a snapshot, without endangering your original data. The software would be operating on the snapshot, not the original logical volume.

You can also use them as alternative versions of a logical volume. You can read and write to a snapshot. A write will change only the snapshot volume, not the original, creating, in effect, an alternate version.

Configuring RAID Devices

RAID is a method of storing data across several disks to provide greater performance and redundancy. In effect, you can have several hard disks treated as just one hard disk by your operating system. RAID then efficiently stores and retrieves data across all these disks,

instead of having the operating system access each one as a separate file system. Lower-level details of storage and retrieval are no longer a concern of the operating system. This allows greater flexibility in adding or removing hard disks, as well as implementing redundancy in the storage system to provide greater reliability. With RAID, you can have several hard disks that are treated as one virtual disk, where some of the disks are used as real-time mirrors, duplicating data. You can use RAID in several ways, depending upon the degree of reliability you need. When you place data on multiple disks, I/O operations can overlap in a balanced way, improving performance. Because having multiple disks increases the mean time between failures (MTBF), storing data redundantly also increases fault tolerance.

RAID can be implemented on a hardware or software level. On a hardware level, you can have hard disks connected to a RAID hardware controller, usually a special PC card. Your operating system then accesses storage through the RAID hardware controller. Alternatively, you can implement RAID as a software controller, letting a software RAID controller program manage access to hard disks treated as RAID devices. The software version lets you use IDE hard disks as RAID disks. Linux uses the MD driver, supported in the 2.4 kernel, to implement a software RAID controller. Linux software RAID supports five levels (linear, 0, 1, 4, 5, and 6), whereas hardware RAID supports many more. Hardware RAID levels, such as 7–10, provide combinations of greater performance and reliability.

TIP *Before you can use RAID on your system, make sure it is supported on your kernel, along with the RAID levels you want to use. If not, you will have to reconfigure and install a RAID module for the kernel. Check the Multi-Driver Support component in your kernel configuration. You can specify support for any or all of the RAID levels.*

Motherboard RAID Support: dmraid

With kernel 2.6, hardware RAID devices are supported with the Device-Mapper Software RAID support tool (**dmraid**), which currently supports a wide range of motherboard RAID devices. Keep in mind that many "hardware" RAID devices are, in effect, really software RAID (fakeraid). Though you configure them in the motherboard BIOS, the drivers operate as software, like any other drivers. In this respect they could be considered less flexible than a Linux software RAID solution, and they could also depend directly on vendor support for any fixes for updates.

The **dmraid** driver will map your system to hardware RAID devices such as those provided by Intel, Promise, and Silicon Magic, and often included on motherboards. The **dmraid** tool uses the device-mapper driver to set up a virtual file system interface, just as is done for LVM drives. The RAID device names will be located in **/dev/mapper**.

You use your BIOS RAID configuration utility to set up your RAID devices as instructed by your hardware documentation. During a Linux installation, the RAID devices are automatically detected and the **dmraid** module is loaded, selecting the appropriate drivers.

With the **dmraid** command you can detect and activate RAID devices. The following command displays your RAID devices:

```
dmraid -r
```

To list currently supported devices, use **dmraid** with the **-l** option.

```
dmraid -l
```

The **dmraid** tool is improved continually and may not work well with some RAID devices.

Linux Software RAID Levels

Linux software RAID can be implemented at different levels, depending on whether you want organization, efficiency, redundancy, or reconstruction capability. Each capability corresponds to different RAID levels. For most levels, the size of the hard disk devices should be the same. For mirroring for RAID 1, disks of the same size are required, and for RAID 5 they are recommended. Linux software RAID supports five levels, as shown in Table 30-2. RAID 5 requires at least three hard drives.

Linear

The *linear* level lets you simply organize several hard disks into one logical hard disk, providing a pool of continuous storage. Instead of being forced to set up separate partitions on each hard drive, in effect you have only one hard drive. The storage is managed sequentially. When one hard disk fills up, the next one is used. In effect, you are *appending* one hard disk to the other. This level provides no recovery capability. If you have a hard disk RAID array containing two 80 GB disks, after you use up the storage on one, you will automatically start on the next.

RAID Level	Capability	Description
Linear	Appending	Treats RAID hard drives as one virtual drive with no striping, mirroring, or parity reconstruction.
0	Striping	Implements disk striping across drives with no redundancy.
1	Mirroring	Implements a high level of redundancy. Each drive is treated as mirror for all data.
5	Distributed parity	Implements data reconstruction capability using parity information. Parity information is distributed across all drives, instead of using a separate drive as in RAID 4. Requires at least three hard drives or partitions.
6	Distributed parity	Implements data reconstruction capability using dual distributed parity information. Dual sets of parity information are distributed across all drives. Can be considered an enhanced form of 5.
Multipath	Multiple access to devices	Supports multiple access to the same device.

TABLE 30-2 Linux Software RAID Levels

RAID 0: Striping

For efficiency, RAID stores data using disk *striping*, where data is organized into standardized stripes that can be stored across the RAID drives for faster access (level 0). RAID 0 also organizes your hard disks into common RAID devices but treats them like single hard disks, storing data randomly across all the disks. If you had a hard disk RAID array containing two 80 GB disks, you could access them as one 160 GB RAID device.

RAID 1: Mirroring

RAID level 1 implements redundancy through *mirroring*. In mirroring, the same data is written to each RAID drive. Each disk has a complete copy of all the data written, so that if one or more disks fail, the others still have your data. Though extremely safe, redundancy can be very inefficient and consumes a great deal of storage. For example, on a RAID array of two 80 GB disk drives, one disk is used for standard storage and the other is a real-time backup. This leaves you with only 80 GB for use on your system. Write operations also have to be duplicated across as many mirrored hard disks as are used by the RAID array, slowing down operations.

RAID 5 and 6: Distributed Parity

As an alternative to mirroring, data can be reconstructed using *parity information* in case of a hard drive crash. Parity information is saved instead of full duplication of the data. Parity information takes up the space equivalent of one drive, leaving most of the space on the RAID drives free for storage. RAID 5 combines both striping and parity (see RAID 4), where parity information is distributed across the hard drives, rather than in one drive dedicated to that purpose. This allows the use of the more efficient access method, striping. With both striping and parity, RAID 5 provides both fast access and recovery capability, making it the most popular RAID level used. For example, a RAID array of four 80 GB hard drives would be treated as one 320 GB hard drive with part of that storage (80 GB) used to hold parity information, leaving 240 GB free. RAID 5 does require at least three hard drives.

RAID 6 operates the same as RAID 5, but it uses dual sets of parity information for the data, providing even greater restoration capability.

RAID 4: Parity

Though it is not supported in some distributions due to overhead costs, RAID 4, like RAID 5, supports a more compressed form of recovery using parity information instead of mirrored data. With RAID 4, parity information is kept on a separate disk, while the others are used for data storage, much like in a linear model.

TIP *Many distributions also allow you to create and format RAID drives during installation. At that time, you can create your RAID partitions and devices.*

Multipath

Though not actually a RAID level, Multipath allows for multiple access to the same device. Should one controller fail, another can be used to access the device. In effect, you have controller-level redundancy. Support is implemented using the **mdadmd** daemon. This is started with the **mdadmd** service script.

```
start mdadmd start
```

RAID Devices and Partitions: md and fd

A RAID device is named an **md** and uses the MD driver. These devices are already defined on your Linux system in the **/etc/dev** directory, starting with **md0**: **/dev/md0** is the first RAID device, and **/dev/md1** is the second, and so on. Each RAID device, in turn, uses hard disk partitions, where each partition contains an entire hard disk. These partitions are usually referred to as RAID disks, whereas a RAID device is an array of the RAID disks it uses.

When creating a RAID partition, you should set the partition type to be **fd**, instead of the 83 for the standard Linux partition. The **fd** type is that used by RAID for automatic detection.

Booting from a RAID Device

Usually, as part of the installation process, you can create RAID devices from which you can also boot your system. Your Linux system will be configured to load RAID kernel support and automatically detect your RAID devices. The boot loader will be installed on your RAID device, meaning on all the hard disks making up that device.

Most Linux distributions do not support booting from RAID 5, only RAID 1. This means that if you want to use RAID 5 and still boot from RAID disks, you will need to create at least two (or more if you want) RAID devices using corresponding partitions for each device across your hard disks. One device would hold your **/boot** partition and be installed as a RAID 1 device. This RAID 1 device would be the first RAID device, **md0**, consisting of the first partition on each hard disk. The second RAID device, **md1**, could then be a RAID 5 device and would consist of corresponding partitions on the other hard disks. Your system could then boot from the RAID 1 device but use the RAID 5 device.

If you do not create RAID disks during installation but instead create them later and want to boot from them, you will have to make sure your system is configured correctly. The RAID devices need to be created with persistent superblocks, and support for the RAID devices has to be enabled in the kernel. On Linux distributions, this support is enabled as a module. Difficulties occur if you are using RAID 5 for your / (root) partition. This partition contains the RAID 5 module, but to access the partition, you have to already load the RAID 5 module. To work around this limitation, you can create a RAM disk in the **/boot** partition that contains the RAID 5 module. Use the **mkinitrd** command to create the RAM disk and the **-with** option to specify the module to include.

```
mkinitrd --preload raid5 --with=raid5 raid-ramdisk 2.6.9-1
```

RAID Administration: mdadm

You use the **mdadm** tool to manage and monitor RAID devices. The **mdadm** tool is an all-purpose means of creating, monitoring, administering, and fixing RAID devices. You can run commands directly to create and format RAID disks. It also runs as a daemon to monitor and detect problems with the devices.

The **mdadm** tool has seven different modes of operation, each with its own set of options, including monitor with the **-f** option to run it as a daemon, or create with the **-l** option to set a RAID level for a disk. Table 30-3 lists the different modes of operation. Check the **mdadm** Man page for a detailed listing of the options for each mode.

Mode	Description
assemble	Assembles RAID array from devices.
build	Builds array without per-device superblocks.
create	Builds array with per-device superblocks.
manage	Manages array devices, adding or removing disks.
misc	Performs specific operations on a device, such as making it read only.
monitor	Monitors arrays for changes and acts on them (used for RAID 1, 4, 5, 6).
grow	Changes array size, as when replacing smaller devices with larger ones.

TABLE 30-3 **Mdadm** Modes

Creating and Installing RAID Devices

If you created your RAID devices and their partitions during the installation process, you should already have working RAID devices. Your RAID devices will be configured in the **/etc/mdadm.conf** file, and the status of your RAID devices will be listed in the **/proc/mdstat** file. You can manually start or stop your RAID devices with the **raidstart** and **mdadm** commands. The **-a** option operates on all of them, though you can specify particular devices if you want.

To create a new RAID device manually for an already-installed system, follow these steps:

- Make sure that your kernel supports the RAID level you want for the device you are creating.
- If you have not already done so, create the RAID disks (partitions) you will use for your RAID device.
- Create your RAID device with **mdadm** command in the build or create mode. The array will also be activated.
- Alternatively, you can configure your RAID device (**/dev/md***n*) in the **/etc/mdadm.conf** file, specifying the RAID disks to use, and then use the **mdadm** command specifying the RAID device to create.
- Create a file system on the RAID device (**mkfs**) and then mount it.

Adding a Separate RAID File System

If you just want to add a RAID file system to a system that already has a standard boot partition, you can dispense with the first RAID 1 partition. Given three hard disks to use for the RAID file system, you just need three partitions, one for each disk, **sda1, sdb1,** and **sdc1**.

```
ARRAY /dev/md0   devices=/dev/sda1,/dev/sdb1  level=1 num-devices=2
```

You then create the array with the following command:

```
mdadm -C /dev/md0 -n3 /dev/sda1 /dev/sdb1 /dev/sdc1 -l5
```

You can then format and mount your RAID device.

Creating Hard Disk Partitions: fd

To add new RAID devices or to create them in the first place, you need to manually create the hard disk partitions they will use and then configure RAID devices to use those partitions. To create a hard disk partition for use in a RAID array, use **fdisk** or **parted** and specify **fd** as the file system type. You invoke **fdisk** or **parted** with the device name of the hard disk you want to create the partition on. Be sure to specify **fd** as the partition type. The following example invokes **fdisk** for the hard disk **/dev/hdc** (the first hard disk on the secondary IDE connection):

```
fdisk /dev/hdc
```

Though technically partitions, these hard disk devices are referred to as disks in RAID configuration documentation and files.

NOTE *You can also use* ***gparted*** *or* ***qtparted*** *to create your hard disk partitions. These tools provide a GUI interface for* ***parted*** *(Applications | System Tools menu).*

Configuring RAID: /etc/mdadm.conf

Once you have your disks, you then need to configure them as RAID devices. RAID devices are configured in the **/etc/mdadm.conf** file, with options as shown in Table 30-4. This file will be used by the **mdadm** command in the create mode to create the RAID device. In the

Directive or Option	Description
DEVICE *devices-list*	Partitions and drives used for RAID devices.
ARRAY	ARRAY configuration section for a particular RAID device.
level=*num*	The RAID level for the RAID device, such as 0, 1, 4, 5, and –1 (linear).
devices=*disk-device-list*	The disk devices (partitions) that make up the RAID array.
num-devices=*count*	Number of RAID devices in an array. Each RAID device section must have this directive. The maximum is 12.
spare-group=*name*	Text name for a spare group, whose devices can be used for other arrays.
auto=*option*	Automatically create specified devices if they do not exist. You can create traditional nonpartitioned (**yes** or **md** option) or the newer partitionable arrays (**mdp** or **part** option). For partitionable arrays the default is 4, which you can change.
super-minor	Minor number of the array superblock, same as md device number.
uuid=*UUID-number*	UUID identifier stored in array superblock, used to identify the RAID array. Can be used to reference an array in commands.
MAILADDR	Monitor mode, mail address where alerts are sent.
PROGRAM	Monitor mode, program to run when events occur.

TABLE 30-4 **mdadm.conf** Options

/etc/mdadm.conf file, you create both DEVICE and ARRAY entries. The DEVICE entries list the RAID devices. The ARRAY entries list the RAID arrays and their options. This example implements a simple array on two disks. Serial ATA drives are used that have a device name prefix **sd** instead of **hd**.

```
DEVICE  /dev/sda1 /dev/sdb1
```

You can list more than one device for a DEVICE entry, as well as have separate DEVICE entries. You can also specify multiple devices using file matching symbols, like *, ?, or []. The following specifies all the partitions on **sda** drive as RAID devices:

```
DEVICE  /dev/sda*  /dev/sdb1
```

For an **ARRAY** entry, you specify the name of the RAID device you are configuring, such as **/dev/md0** for the first RAID device. You then add configuration options such as **devices** to list the partitions that make up the array, **level** for the RAID level, and **num-devices** for the number of devices.

```
ARRAY /dev/md0   devices=/dev/sda2,/dev/sdb1, /dev/sdc1  level=5 num-devices=3
```

The preceding example configures the RAID array **/dev/md0** as three disks (partitions) using **/dev/sda1**, **/dev/sdb1**, and **/dev/sdc1**, and is configured for RAID 5 (**level=5**).

Creating a RAID Array

You can create a RAID array either using options specified with the **mdadm** command or using configurations listed in the **/etc/mdadm.conf** file. Use of the **/etc/mdadm.conf** file is not required, though it does make RAID creation more manageable, especially for large or complex arrays. Once you have created your RAID devices, your RAID device will be automatically activated. The following command creates a RAID array, **/dev/md1**, using two devices, **/dev/sda2**, **/dev/sdb1**, and **/dev/sdc1**, at level 5.

```
mdadm --create /dev/md1 --raid-devices=3 /dev/sda1 /dev/sdb1 /dev/sdc1 --level=5
```

Each option has a corresponding short version, as shown in Table 30-5. The same command is shown here with single-letter options.

```
mdadm -C /dev/md1 -n3 /dev/sda1 /dev/sdb1 /dev/sdc1 -l5
```

If you have configured your RAID devices in the **/etc/mdadm.conf** file, you use the **mdadm** command in the create mode to create your RAID devices. **mdadm** takes as its argument the name of the RAID device, such as **/dev/md0** for the first RAID device. It then locates the entry for that device in the **/etc/mdadm.conf** file and uses that configuration information to create the RAID file system on that device. You can specify an alternative configuration file with the **-c** option, if you wish. **mdadm** operates as a kind of **mkfs** command for RAID devices, initializing the partitions and creating the RAID file systems. Any data on the partitions making up the RAID array will be erased.

```
mdadm  --create /dev/md0
```

mdadm –create Option	Description
`-n --raid-devices`	Number of RAID devices
`-l --level`	RAID level
`-C --create`	Create mode
`-c --chunk`	Chunk (stripe) size in powers of 2; default is 64 KB
`-x --spare-devices`	Number of spare devices in the array
`-z --size`	Size of blocks used in devices, by default set to the smallest device if not the same size
`-p --parity`	The parity algorithm; left-symmetric is used by default

TABLE 30-5 The mdadm –create Options

Creating Spare Groups

Linux software RAID now allows RAID arrays to share their spare devices. This means that if arrays belong to the same spare group, a device that fails in one array can automatically use the spare in another array. Spare devices from any array can be used in another as needed. You set the spare group that an array belongs to with the **`--spare-group`** option. The **`mdadm`** monitoring mode will detect a failed device in an array and automatically replace it with a spare device from arrays in the same spare group. The first command in the next example creates a spare drive called **/dev/hdd1** for the **/dev/md0** array and labels it **mygroup**. In the second command, array **/dev/md1** has no spare drive but belongs to the same spare group as array **/dev/md0**. Should a drive in **/dev/md1** fail, it can automatically use the spare device, **/dev/hdd1**, from **/dev/md0**. The following code lines are really two lines, each beginning with **`mdadm`**:

```
mdadm --create /dev/md0 --raid-devices=3 /dev/hda1 /dev/hdc1 -x
       /dev/hdd1 --level=1 --spare-group=mygroup
mdadm --create /dev/md1 --raid-devices=2 /dev/hda2 /dev/hdc2 --level=1
       --spare-group=mygroup
```

Creating a File System

Once the RAID devices are activated, you can create file systems on the RAID devices and mount those file systems. The following example creates a standard Linux file system on the **/dev/md0** device:

```
mkfs.ext3 /dev/md0
```

In the following example, the user then creates a directory called **/myraid** and mounts the RAID device there:

```
mkdir /myraid
mount /dev/md0 /myraid
```

If you plan to use your RAID device for maintaining your user directories and files, you mount the RAID device as your **/home** partition. Such a mounting point might normally be

used if you created your RAID devices when installing your system. To transfer your current home directories to a RAID device, first back them up on another partition, and then mount your RAID device, copying your home directories to it.

Managing RAID Arrays

You can manage RAID arrays with the **mdadm** manage mode operations. In this mode you can add or remove devices in arrays, as well as mark ones as failed. The **--add** option lets you add a device to an active array, essentially a hot swap operation.

```
mdadm /dev/md0 =-add /dev/sdd1
```

To remove a device from an active array, you first have to mark it as failed with the **--fail** option and then remove it with **--remove**.

```
mdadm /dev/md0 --fail /dev/sdb1 --remove /dev/sdb1
```

Starting and Stopping RAID Arrays

To start an already existing RAID array, you use **mdadm** with the assemble mode (newly created arrays are automatically started). To do so directly on the command line requires that you also know what devices make up the array, listing them after the RAID array.

```
mdadm -A /dev/md1 /dev/sda2  /dev/sdb1
```

It is easier to configure your RAID arrays in the **/etc/mdadm.conf** file. With the scan option, **-s**, **mdadm** will then read array information from the **/etc/mdadm.conf** file. If you do not specify a RAID array, all arrays will be started.

```
mdadm -As /dev/md0
```

To stop a RAID array, you use the **-S** option.

```
mdadm -S /dev/md0
```

Monitoring RAID Arrays

As a daemon, **mdadm** is started and stopped using the **mdmonitor** service script in **/etc/init.d**. This invokes **mdadm** in the monitor mode, detecting any problems that arise and logging reports as well as taking appropriate action.

```
service mdadm start
```

You can monitor devices directly by invoking **mdadm** with the monitor mode.

```
mdadm --monitor /dev/md0
```

Monitor-related options can be set in the **/etc/mdadm.conf** file. MAILADDR sets the mail address where notification of RAID events are sent. PROGRAM sets the program to use if events occur.

If you decide to change your RAID configuration or add new devices, you first have to deactivate your currently active RAID devices. To deactivate a RAID device, you use the

mdadm command in the misc mode. Be sure to close any open files and unmount any file systems on the device first.

```
umount /dev/md0
mdadm -S /dev/md0
```

Configuring Bootable RAID

Bootable RAID devices use RAID level 0 or 1. In the following example, the first array holds only the boot partition and uses RAID level 1, whereas the second array uses three partitions and is set to RAID level 5.

```
ARRAY /dev/md0   devices=/dev/sda1  level=1 num-devices=1
ARRAY /dev/md1   devices=/dev/sda2,/dev/sdb1/dev/sdc1  level=5 num-devices=3
```

The preceding example configures the RAID array **/dev/md0** as a RAID 1 (**level=1**) device. Two disks (partitions) make up the second RAID array, **/dev/md1** using **/dev/sda2** and **/dev/sdb1**. It is configured for RAID 5 (**level=5**).

You can create a RAID array either using options specified with the **mdadm** command or using configurations listed in the **/etc/mdadm.conf** file. Use of the **/etc/mdadm.conf** file is not required, though it does make RAID creation more manageable, especially for large or complex arrays. Once you have created your RAID devices, your RAID device will be automatically activated. The following command creates a RAID array, **/dev/md1**, using two devices, **/dev/sda2**, **/dev/sdb1**, and **/dev/sdc1**, at level 5.

```
mdadm --create /dev/md1 --raid-devices=2 /dev/sda2 /dev/sdb1 --level=1
```

Each option has a corresponding short version, as shown in Table 30-5. The same command is shown here with single-letter options.

```
mdadm -C /dev/md1 -n3 /dev/sda2 /dev/sdb1/dev/sdc1 -l5
```

For the first array in the previous example you use:

```
mdadm -C /dev/md0 -n2 /dev/sdaa -l1
```

To more efficiently use space, you could use corresponding hard disk partitions as described in the next section.

Corresponding Hard Disk Partitions

The term *device* can be confusing, because it is also used to refer to the particular hard disk partitions that make up a RAID device. In fact, a software RAID device is an array of hard disk partitions, where each partition can, but need not, take up an entire hard disk. In that case, you can think of a RAID device as consisting of a set (array) of hard disks (devices). In practice, the hard disks in your RAID configuration normally contain several corresponding hard disk partitions, each set having the same size. Each set of corresponding partitions makes up a RAID device, so you can have several RAID devices using the same set of hard disks. This is particularly true for Linux partition configurations, where different system directories are placed in their own partitions. For example, **/boot** could be in one partition, **/home** in another, and **/** (the root) in yet another partition. This approach also lets you set up

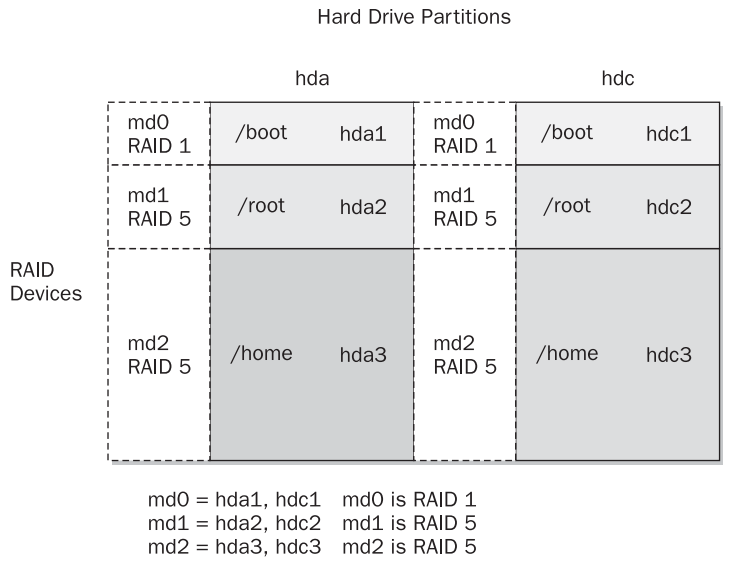


FIGURE 30-2 RAID devices (RAID 5 will require at least three partitions)

a RAID5 implementation using just two physical hard drives. In effect you could have a raid device using four partitions on two hard drives. You would, though, lose much of the reliability of a basic three or more drive RAID5 implementation.

To set up RAID devices so that you have separate partitions for **/boot**, **/home**, and **/** (root), you need to create three different RAID devices, say **md0** for **/boot**, **md1** for **/home**, and **md2** for the root. If you have two hard disks, for example **hda** and **hdc**, each would have three partitions, **/boot**, **/home**, and **/**. The first RAID device, **md0**, would consist of the two **/boot** partitions, the one on **hda** and the one on **hdc**. Similarly, the second RAID device, **md1**, would be made up of the two root partitions, **/**, the one on **hda** and the other on **hdc**. **md3** would consist of the **/home** partitions on **hda** and **hdc** (see Figure 30-2).

When you create the partitions for a particular RAID device, it is important to make sure that each partition has the same size. For example, the **/** partition used for the **md0** device on the **hda** disk must have the same size as the corresponding **md0** partition on the **hdc** disk. So if the **md1** partition on **hda** is 20 GB, then its corresponding partition on **hdc** must also be 20 GB. If **md2** is 100 GB on one drive, its corresponding partitions on all other drives must also be 100 GB.

RAID Example

Figure 30-2 shows a simple RAID configuration with three RAID devices using corresponding partitions on two hard disks for **/boot**, **/** (root), and **/home** partitions. The boot partition is configured as a RAID 1 device because systems can be booted only from a RAID 1 device, not RAID 5. To simplify this example, only two partition are used for RAID5. But keep in mind that RAID5 requires at least three partitions.

You can set up such a system during installation, selecting and formatting your RAID devices and their partitions using Disk Druid. The steps described here assume you have

your system installed already on a standard IDE drive and are setting up RAID devices on two other IDE disk drives. You can then copy your file from your standard drive to your RAID devices.

First you create the hard disk partitions using a partition tool like **parted** or **fdisk**. Then, configure the three RAID devices in the **/etc/mdadm.conf** file.

```
DEVICE  /dev/hda1 /dev/hda2 /dev/hda3 /dev/hdc1 /dev/hdc2 /dev/hdc3

ARRAY /dev/md0   devices=/dev/hda1,/dev/hdc1  level=1 num-devices=2
ARRAY /dev/md1   devices=/dev/hda2,/dev/hdc2  level=5 num-devices=2
ARRAY /dev/md2   devices=/dev/hda3,/dev/hdc3 level=5 num-devices=2
```

Next, create your RAID devices with **mdadm**, which will then be automatically activated.

```
mdadm --create /dev/md0 /dev/md1 /dev/md2
```

Create your file systems on the RAID devices.

```
mkfs.ext3 md0 md1 md2
```

You can then migrate the **/boot**, **/**, and **/home** files from your current hard disk to your RAID devices. Install your boot loader on the first RAID device, **md0**, and load the root file system from the second RAID device, **md1**.

Alternatively, you can first create the arrays with the **mdadm** command and then generate the ARRAY entries for an **/etc/mdadm.conf** file from the created RAID information to later manage your arrays, adding or removing components. The following commands create the three RAID devices in the previous example:

```
mdadm --create /dev/md0 --raid-devices=2 /dev/hda1 /dev/hdc1 --level=0
mdadm -C /dev/md2 -n3 /dev/hda2 /dev/hdc2 -l5
mdadm -C /dev/md2 -n3 /dev/hda3 /dev/hdc3 -l5
```

You can then generate the ARRAY entries for the **/etc/mdadm.conf** file directly using the following command. You will still have to edit **mdadm.conf** and add the DEVICE entries as well as the monitoring entries, like MAILADDR.

```
mdadm --detail --scan > /etc/mdadm.conf
```

The previous example could also be implemented using RAID 0 to boot and RAID1 for **md1** and **md2**, making the **hdc** disk function as a mirror disk for both **md0** and **md1** arrays.

This page intentionally left blank

CHAPTER 31

Devices and Modules

All devices, such as printers, terminals, and CD-ROMs, are connected to your Linux operating system through special files called *device files*. Such files contain all the information your operating system needs to control the specified device. This design introduces great flexibility. The operating system is independent of the specific details for managing a particular device; the specifics are all handled by the device file. The operating system simply informs the device what task it is to perform, and the device file tells it how. If you change devices, you have to change only the device file, not the whole system.

To install a device on your Linux system, you need a device file for it, software configuration such as that provided by a configuration tool, and kernel support—usually supplied by a module or support that is already compiled and built into the kernel. Device files are not handled in a static way. They are dynamically generated as needed by udev and managed by HAL. Previously a device file was created for each possible device, leading to a very large number of device files in the **/etc/dev** directory. Now, your system will detect only those devices it uses and create device files for those only, giving you a much smaller listing of device files. Both udev and HAL are hotplug systems, with udev used for creating devices and HAL designed for providing information about them, as well as managing the configuration for removable devices such as those with file systems such as those for USB card readers and CD-ROMs.

Managing devices is at the same time easier but much more complex. You now have to use udev and HAL to configure devices, though much of this is now automatic. Device information is maintained in a special device file system called **sysfs** located at **/sys**. This is a virtual file system like **/proc** and is used to keep track of all devices supported by the kernel. Several of the resources you may need to consult and directories you may have to use are listed in Table 31-1.

The sysfs File System: /sys

The system file system is designed to hold detailed information about system devices. This information can be used by hotplug tools like udev to create device interfaces as they are needed. Instead of having a static and complete manual configuration for a device, the **sysfs** system maintains configuration information about the device, which is then used as needed by the hotplugging system to create device interfaces when a device is attached to the system.

Resource	Description
/etc/sysconfig/hwconf	Hardware configuration and listing for your system
/sys	The sysfs file system listing configuration information for all the devices on your system
/proc	An older process file system listing kernel information, including device information
kernel.org/pub/linux/docs/device-list/devices.txt	Linux device names
kernel.org/pub/linux/utils/kernel/hotplug/udev.html	The udev website
/etc/udev	The udev configuration directory
freedesktop.org/wiki/Software/hal	The HAL website
/etc/hal	The HAL configuration directory
/usr/share/hal/fdi	The HAL device information files, for configuring HAL information support and policies
/etc/hal/fdi	The HAL system administrator's device information files

TABLE 31-1 Device Resources

More and more devices are now removable, and many are meant to be attached temporarily (cameras, for example). Instead of maintaining separate static and dynamic methods for configuring devices, Linux distributions make all devices structurally hotplugged.

The **sysfs** file system is a virtual file system that provides the a hierarchical map of your kernel-supported devices such as PCI devices, buses, and block devices, as well as supporting kernel modules. The **classes** subdirectory will list all your supported devices by category, such as net and sound devices. With **sysfs** your system can easily determine the device file a particular device is associated with. This is very helpful for managing removable devices as well as dynamically managing and configuring devices as HAL and udev do. The **sysfs** file system is used by udev to dynamically generate needed device files in the **/dev** directory, as well as by HAL to manage removable device files as needed. The **/sys** file system type is **sysfs**. The **/sys** subdirectories organize your devices into different categories. The file system is used by **systool** to display a listing of your installed devices. The tool is part of the **sysfsutils** package. The following example will list all your system devices.

```
systool
```

Like **/proc**, the **/sys** directory resides only in memory, but it is still mounted in the **/etc/fstab** file.

```
none    /sys        sysfs     defaults   0       0
```

File	Description
/proc/devices	Lists the device drivers configured for the currently running kernel
/proc/dma	Displays the DMA channels currently used
/proc/interrupts	Displays the IRQs (interrupts) in use
/proc/ioports	Shows the I/O ports in use
/proc/pci	Lists PCI devices
/proc/asound	Lists sound devices
/proc/ide	Directory for IDE devices
/proc/net	Directory for network devices

TABLE 31-2 **/proc** Device Information Files

The proc File System: /proc

The **/proc** file system (see Chapter 29) is an older file system that was used to maintain information about kernel processes, including devices. It maintains special information files for your devices, though many of these are now supported by the **sysfs** file system. The **/proc/devices** file lists your installed character and block devices along with their major numbers. IRQs, DMAs, and I/O ports currently used for devices are listed in the **interrupts**, **dma**, and **ioports** files, respectively. Certain files list information covering several devices, such as **pci**, which lists all your PCI devices, and **sound**, which lists all your sound devices. The **sound** file lists detailed information about your sound card. Several subdirectories, such as **net**, **ide**, and **scsi**, contain information files for different devices. Certain files hold configuration information that can be changed dynamically, such as the IP packet forwarding capability and the maximum number of files. You can change these values with the **sysctl** tool (Kernel Tuning in the System Tools menu) or by manually editing certain files. Table 31-2 lists several device-related **/proc** files.

udev: Device Files

Devices are now *hotpluggable,* meaning they can be easily attached and removed. Their configuration is dynamically detected and does not rely on manual administrative settings. The hotplug tool used to detect device files is udev, user devices. Each time your system is booted, udev automatically detects your devices and generates device files for them in the **/etc/dev** directory. This means that the **/etc/dev** directory and its files are recreated each time you boot. It is a dynamic directory, no longer static. udev uses a set of rules to direct how device files are to be generated, including any corresponding symbolic links. These are located in the **/etc/udev/rules.d** file. You can find out more about udev at **kernel.org/pub/linux/utils/kernel/hotplug/udev.html**.

As part of the hotplug system, udev will automatically detect kernel devices that are added or removed from the system. When the device interface is first created, its corresponding **sysfs** file is located and read, determining any additional attributes such as serial numbers and device major and minor numbers that can be used to uniquely identify

the device. These can be used as keys in udev rules to create the device interface. Once the device is created, it is listed in the udev database, which keeps track of currently installed devices.

If a device is added, udev is called by hotplug. It checks the **sysfs** file for that device for the major and minor numbers, if provided. It then uses the rules in its rules file to create the device file and any symbolic links to create the device file in **/dev**, with permissions specified for the device in the udev permissions rules. Once the device file is created, udev runs the programs in **/etc/dev.d**.

NOTE *When the system starts, it invokes **/sbin/udevstart**, which runs udev and creates all the kernel devices making device files in the **/dev** directory.*

As **/etc/dev** is now dynamic, any changes you make manually to the **/etc/dev** directory will be lost when you reboot. This includes the creation of any symbolic links such as **/dev/cdrom** that many software applications use. Instead, such symbolic links have to be configured in the udev rules files, located in the **/etc/udev/rules.d** directory. Default rules are already in place for symbolic links, but you can create rules to add your own.

udev Configuration

The configuration file for udev is **/etc/udev/udev.conf**. Here are set global udev options such as the location of the udev database; the defaults for device permissions, owner, and group; and the location of udev rules files. The udev tool uses the udev **rules.d** file to dynamically create your device files. Be very careful in making any changes, particularly to rules file locations. Support for all devices on your system relies on these rules. The default **udev.conf** file is shown here:

```
# udev.conf
# The main config file for udev
#
# This file can be used to override some of udev's default values
# for where it looks for files, and where it places device nodes.
#
# WARNING: changing any value, can cause serious system breakage!

# udev_root - where in the filesystem to place the device nodes
udev_root="/dev/"

# udev_db - The name and location of the udev database.
udev_db="/dev/.udev.tdb"

# udev_rules - The name and location of the udev rules file
udev_rules="/etc/udev/rules.d/"

#udev_permissions - The name,location of the udev permission file
udev_permissions="/etc/udev/permissions.d/"

# default_mode - set the default mode for all nodes that have no
# explicit match in the permissions file
default_mode="0600"
```

```
#default_owner - set the default owner for all nodes that have no
#  explicit match in the permissions file
default_owner="root"

#default_group - set the default group for all nodes that have no
# explicit match in the permissions file
default_group="root"

# udev_log - set to "yes" if you want logging, else "no"
udev_log="no"
```

The location of the device files directory is set by **udev_root** to **/dev**. This is the official device directory on Linux systems and should never be changed. The **udev_rules** file specifies where the rules files that udev uses to generate the device files are located. The **udev_log** option lets you turn logging on and off, which is useful for detecting errors. The **udev_permissions** option specifies the location of the permission files that hold the permissions to be applied to certain devices. Default permissions are already listed for you. In the case where a device has no permission listed in a permission file, the defaults set by **default_mode**, **default_owner** and **default_group** are used. These function more like fail-safes. A device permission should be set in a udev permission file.

Device Names and udev Rules: /etc/udev/rules.d

The name of a device file is designed to reflect the task of the device. Printer device files begin with **lp** for "line print." Because you can have more than one printer connected to your system, the particular printer device files are distinguished by two or more numbers or letters following the prefix **lp**, such as **lp0**, **lp1**, **lp2**. The same is true for terminal device files, which begin with the prefix **tty**, for "teletype;" they are further distinguished by numbers or letters such as **tty0**, **tty1**, **ttyS0**, and so on. You can obtain a complete listing of the current device filenames and the devices for which they are used from the **kernel.org** website at **kernel.org/pub/linux/docs/device-list/devices.txt**.

With udev, device names are determined dynamically by rules listed in the udev rules files. These are located in **/etc/udev/rules.d**. The rules files that you will find in this directory are generated by your system during installation. You should never edit them. If you need to add rules of your own, you should create your own rules file. The rules files are named, beginning with a number to establish priority. They are read sequentially, with the first rules overriding any conflicting later ones. All rules files have a **.rules** extension. Rules can be organized in three general categories: names, permissions/ownership, and symbolic links. In addition, there are various specialized categories for certain kinds of devices like faxes and certain mice. On Red Hat, Fedora, and similar distributions all three categories are included in a primary rules file called **50-udev.rules**. Here you will find the rules for most of your system devices. On Debian, Ubuntu, and similar distributions, the general cateoriies are held in separate file: **20-names.rules**, **40-permissions.rules**, and **60-symlink.rules.**

Other rules files may be set up for more specialized devices like **60-rules-libsane** for scanners, **60-rules-libmtp** for music players, **60-pcmcia.rules** for PCMCIA devices, and **90-alsa.rules** for the sound driver.

The rules files already present in the **rules.d** directory have been provided for your Linux distribution and are designed specifically for it. You should never modify these rules. To customize your setup, create your own separate rules files in **/etc/udev/rules.d**. In your

rules file you normally define only symlinks, using SYMLINK fields alone, as described in the following sections. These set up symbolic links to devices, letting you access them with other device names. NAME fields are used to create the original device interface, a task usually left to udev itself.

Each line maps a device attribute to a device name, as well as specifying any symbolic names (links). Attributes are specified using keys, of which there may be more than one. If all the keys match a device, then the associated name is used for it and a device file of that name will be generated. An assignable key, like NAME for the device name or SYMLINK for a symbolic name, is use to assign the matched value. Instead of listing a device name, a program or script may be specified instead to generate the name. This is often the case for DVD/CD-ROM devices, where the device name could be a dvdwriter, cdwriter, cdrom, or dvdrom.

The rules consist of a comma-separated list of fields. A field consists of a matching or assignable key. The matching keys use the **==** and **!=** operators to compare for equality and inequality. The *, **?**, and **[]** operators can match any characters, any single character, or a class of characters, just as in the shell. The assignble keys can use the **=**, **+=**, and **:=** operators to assign values. The **=** operator assigns a single value, the **+=** appends the value to those already assigned, and the **:=** operator makes an assignment final, preventing later changes. The assignable keys also support the udev keys listed in Table 31-3. Check the udev Man page for detailed descriptions.

The key fields such as KERNEL support pattern matching to specify collections of devices. For example, mouse* will match all devices beginning with the pattern "mouse". The following field uses the KERNEL key to match on all mouse devices as listed by the kernel:

```
KERNEL=="mouse*"
```

The next key will match on all printer devices numbered **lp0** through **lp9**. It uses brackets to specify a range of numbers or characters, in this case 0 through 9, **[0-9]**:

```
KERNEL=="lp[0-9]*"
```

The NAME, SYMLINK, and PROGRAM fields support string substitution codes similar to the way print codes work. Such a code is preceded by a % symbol. The code allows several possible devices and names to be referenced in the same rule. Table 31-4 lists the supported codes.

The udev Man page provides many examples of udev rules using various fields. The **50-udev.rules** and **20-names.rules** files hold rules that primarily use KERNEL keys to designate devices. The KERNEL key is followed by either a NAME key to specify the device filename or a SYMLINK key to set up a symbolic link for a device file. The following rule uses the KERNEL key to match on all mouse devices as listed by the kernel. Corresponding device names are placed in the **/dev/input** directory, and the name used is the kernel name for the device (%**k**):

```
KERNEL=="mouse*",  NAME="input/%k"
```

This rule uses both a BUS key and a KERNEL key to set up device files for USB printers, whose kernel names will be used to create device files in **/dev/usb**:

```
BUS=="usb", KERNEL=="lp[0-9]*", NAME="usb/%k"
```

Matching Keys	Description
ACTION	Match the event action.
DEVPATH	Match the device path.
ENV{*key***}**	Match an environment variable value.
BUS	Match the bus type of the device. (The sysfs device bus must be able to be determined by a "device" symlink.)
DRIVER	Match the device driver name.
ID	Match the device number on the bus, for instance, the PCI bus ID.
KERNEL	Match the kernel device name.
PROGRAM	Use an external program to determine the device. This key is valid if the program returns successful. The string returned by the program may be additionally matched with the RESULT key.
RESULT	Match the returned string of the last PROGRAM call. This key may be used in any following rule after a PROGRAM call.
SUBSYSTEM	Match the device subsystem.
SYSFS{*filename***}**	Match the sysfs device attribute, for instance, a label, vendor, USB serial number, SCSI UUID, or file system label.
Assignable Keys	**Description**
NAME	The name of the node to be created, or the name the network interface should be renamed to.
OWNER, GROUP, MODE	The permissions for the device.
PLACE	Match the location on the bus, such as the physical port of a USB device.
ENV{*key***}**	Export variable to environment.
IMPORT{*type***}**	Import results of a program, contents of a text file, or stored keys in a parent device. The type can be **program**, **file**, or **parent**.
SYMLINK	The name of the symbolic link (symlink) for the device.
RUN	Add program to list of programs to be run by device.

TABLE 31-3 udev Rule keys

Symbolic Links

Certain device files are really symbolic links bearing common device names that are often linked to the actual device file used. A *symbolic link* is another name for a file that is used like a shortcut, referencing that file. Common devices like printers, CD-ROM drives, hard drives, SCSI devices, and sound devices, along with many others, will have corresponding symbolic links. For example, a **/dev/cdrom** symbolic link links to the actual device file used for your CD-ROM. If your CD-ROM is an IDE device, it may use the device file **hdc**. In this

Substitution Code	Description
%n	The kernel number of the device
%k	The kernel name for the device
%M	The kernel major number
%m	The kernel minor number
%p	The device path
%b	The device name matched from the device path
%c	The string returned by a PROGRAM field (can't be used in a PROGRAM field)
%s {*filename*}	Content of sysfs attribute
%E{*key*}	Value of environment variable
%N	Name of a temporary device node, to provide access before real node is created
%%	Quotes the % character in case it is needed in the device name

TABLE 31-4 udev Substitution Codes

case, **/dev/cdrom** is a link to **/dev/hdc**. In effect, **/dev/cdrom** is another name for **/dev/hdc**. Serial ATA DVD/CD drives will be linked to **scd** devices, such as **scd0** for the first Serial ATA CD/DVD drive. If your drive functions both as a CD and DVD writer and reader, you will have several links to the same device. In this case the links **cdrom**, **cdrw**, **cdwriter**, **dvd**, **dvdrw**, **dvdwriter** will all link to the same CD/DVD RW-ROM device.

A **/dev/modem** link file also exists for your modem. If your modem is connected to the second serial port, its device file will be **/dev/ttyS1**. In this case, **/dev/modem** will be a link to that device file. Applications can then use **/dev/modem** to access your modem instead of having to know the actual device file used. Table 31-5 lists commonly used device links.

Symbolic links are created by udev using the SYMLINK field. The symbolic links for a device can be listed either with the same rule creating a device file (NAME key) or in a separate rule that will specify only a symbolic link. The inclusion of the NAME key does not have to be specific if the default device name is used. The + added to the = symbol will automatically create the device with the default name, not requiring an explicit NAME key in the rule. The following rule is for parallel printers. It includes both the default name and implied NAME key creating the device (+) and a symbolic link, **par**. The %**n** will add a number to the symbolic link like **par1**, **par2**, and so on.

```
KERNEL=="lp[0-9]*",      SYMLINK+="par%n"
```

Should you want to create more than one symbolic link for a device, you can list them in the SYMLINK field. The following creates two symbolic links, one **cdrom** and another named **cdrom-** with the kernel name attached (%**k**).

```
SYMLINK+="cdrom cdrom-%k"
```

Link	Description
cdrom	Link to your CD-ROM device file, set in **/etc/udev/rules.d**
dvd	Link to your DVD-ROM device file, set in **/etc/udev/rules.d**
cdwriter	Link to your CD-R or CD-RW device file, set in **/etc/udev/rules.d**
dvdwriter	Link to your DVD-R or DVD-RW device file, set in **/etc/udev/rules.d**
modem	Link to your modem device file, set in **/etc/udev/rules.d**
floppy	Link to your floppy device file, set in **/etc/udev/rules.d**
tape	Link to your tape device file, set in **/etc/udev/rules.d**
scanner	Link to your scanner device file, set in **/etc/udev/rules.d**
mouse	Link to your mouse device file, set in **/etc/udev/rules.d**
tape	Link to your tape device file, set in **/etc/udev/rules.d**

TABLE 31-5 Device Symbolic Links

If you decide to set up a separate rule that specifies just a symbolic link, the symbolic link will be kept on a list awaiting the creation of its device. This also allows you to add other symbolic links for a device in other rules files. This situation can be confusing because symbolic links can be created for devices that are not yet generated. The symbolic links will be defined and held until needed, when the device is generated. This is why you can have many more SYMLINK rules than NAME rules in udev that actually set up device files. In the case of removable devices, they will not have a device name generated until they are connected.

In the **50-rules.udev** or **60-symlink.rules** files you will find several SYMLINK rules for optical devices, one of which is shown here, using the implied default name:

```
KERNEL=="scd[0-9]*",          SYMLINK+="cdrom cdrom-%k"
```

In most cases, you will only need symbolic links for devices, using the official symbolic names. Most of these are already defined for you. Should you need to create just a symbolic link, you can create a SYMLINK rule for it. However, a new SYMLINK rule needs to be placed before the name rules that name that device. The SYMLINK rules for a device are read by udev and kept until a device is named. Then those symbolic names can be used for that device. You can have as many symbolic links for the same device as you want, meaning that you could have several SYMLINK rules for the same device. When the NAME rule for the device is encountered, the previous SYMLINK keys are simply appended.

Most standard symbolic names are already defined in the **50-udev.rules** or **60-symlink .rules** file (depending on the distribution), such as **audio** for the audio device. In the following example, the device is referenced by its KERNEL key and the symbolic link is applied with SYMLINK key. This is only a SYMLINK rule. The NAME key is implied:

```
KERNEL=="audio0", SYMLINK+="audio"
```

Program Fields, IMPORT{program} keys, and /lib/udev

Several kinds of devices use special scripts to determine the name to be used for the device. This is particularly true of CD/DVD readers or writers, for which there can be multiple devices of very different types, such as CD-ROMs, DVD-RWs, or CD-RWs. The symbolic link used can be **cdwriter** for a CD-RW, **cdrom** for a CD-ROM, or even **dvd** for a DVD-ROM. To determine the correct symbolic link, a program is invoked to determine the device. Many of these programs are specialized programs and scripts kept in the **/lib/udev** directory, such as **`check-cdrom.sh`**, which determines the CD-ROM type. A program can be invoked either with the **PROGRAM** key or the **IMPORT{program}** key. The **PROGRAM** key will run the program and return 0 if successful. Returned values can be held in the **RESULTS** key. The **IMPORT{program}** key returns assinable values.

On Red Hat, Fedora, and simiar distributions, the **`check-cdrom.sh`** script is used to see if an IDE device (hd) is DVD. In the following rule, the script is passed the kernel name and the DVD parameter and will return a positive value if the device is a DVD-ROM. It is then assigned a **dvd***n* symbolic link, as in **dvd1**. Two other keys are used in this example. The BUS key checks to see if the device is an IDE CD-ROM, and **ATTRS{removable}=="1"** confirms whether it is removable. The following lines are really one line.

```
KERNEL=="hd[a-z]", BUS=="ide", ATTRS{removable}=="1", PROGRAM=="check-
cdrom.sh %k DVD", SYMLINK+="dvd dvd-%k"
```

The following rule is used for simple CD-ROMs. The ATTRS field will return the name of the IDE device in the **sys** file system and its output is tested to see if it is a CD-ROM. Then a symbolic link is assigned, as in **cdrom** or **cdrom-**, with its kernel name attached. The following lines are one line.

```
KERNEL=="hd[a-z]", BUS=="ide", ATTRS{media}== cdrom, SYMLINK+="cdrom cdrom-%k"
```

You can use the **IMPORT(program)** key to import run an external program and import its results direcly as an assigned value. On Debian, the program used for detecting DVD/CD drive devices is **cdrom_id**.

```
 KERNEL=="sr[0-9]*|hd[a-z]*|pcd[a-z]*", IMPORT(program)="cdrom_id –export
$tempnode"
```

On Debian, separate symlink and permission rules are used. The symlink rules use the **ENV** key to check for a value in the associated cdrom_id variables such as **ID_CDROM** for CDROM devices.

```
ENV{ID_CDROM}== "?*" ,   SYMLINK+="cdrom"
```

For permissions, CDROMs are assigned to the cdrom group.

```
ENV{ID_CDROM}== "?*" ,   GROUP+="cdrom"
```

Creating udev Rules

Default rules for your devices are placed by udev in the **/etc/udev/rules.d/50-udev.rules** or **20-names.rules** files. You should never edit this file, though you can check it to see how

device naming is handled. This file will create the device files using the official kernel names. These names are often referenced directly by applications that expect to find devices with these particular names, such as **lp0** for a printer device.

If you want to create rules of your own, you should place them in a separate rules file. The NAME rules that name devices are read lexically, where the first NAME rule will take precedence over any later ones. Only the first NAME rule for a device will be used. Later NAME rules for that same device will be ignored. Keep in mind that a SYMLINK rule with a += includes a NAME rule for the default device, even though the NAME field is not explicitly shown. Check **reactivated.net/writing_udev_rules.html** for a tutorial on writing udev rules.

Since rules are being created that are meant to replace the default rules, they would have to be run first. To do this, you place them in a rules file that begins with a very low number, say 10. Such a rules file would be executed before the **50-udev.rules** or **20-names.rules** files, which holds the default rules. Rules files are read in lexical order, with the lower numbers read first. You can create a file called **10-user.rules** in the **/etc/udev/rules.d** directory in which you place your own rules. Conversely, if you want rules that will run only if the defaults fail for some reason, you can use a rules file numbered after 50, such as **90-mydefaults.rules**.

The upcoming section "Persistent Names: udevinfo," describes how to create a canon-pr rule to replace the default printer rule for that printer. The new user canon-pr rule would be placed in a **10-user.rules** file to be executed before the printer rules in the **50-udev.rules** file or **20-names.rules** files, thereby taking precedence. The default printer rule in the **50-udev .rules** and **20-names.rules** files (shown here) would not be applied to the Canon printer.

```
BUS="usb", KERNEL="lp[0-9]*", NAME="usb/%k"
```

SYMLINK Rules

In most cases, you will only need to create symbolic links for devices, using the official name. You can also create rules that just create symbolic links. However, these need to be placed before the name rules that name the devices. These SYMLINK rules are read by udev and kept until a device is named, then all the symbolic names will be used for that device. You can have as many symbolic links as you want for the same device, meaning that you can have several SYMLINK rules for the same device. When the NAME rule for the device is encountered, the previous SYMLINK keys are simply appended.

Most standard symbolic names are already defined in the **50-udev.rules** and **60-symlink. rules** files, such as audio for the audio device. In the following example, the device is referenced by its KERNEL key and the symbolic link is applied with SYMLINK field. The NAME field for the default device is implied:

```
KERNEL=="audio0",  SYMLINK+="audio"
```

If you always know the name for a device, you can easily add a SYMLINK rule. For example, if you know your DVD-ROM is attached to the first secondary IDE connection (**hdc**), you can create a symbolic name of your own choosing with a SYMLINK rule. In the next example a new symbolic link, **mydvdrom**, is created for the DVD-ROM on the **/dev/hdc** device.

```
KERNEL=="hdc",     SYMLINK="mydvdrom"
```

For a SYMLINK rule to be used, it must occur before a NAME rule that names the device. You should place these rules in a file that will precede the **50-udev.rules** or **60-symlink.rules** files, such as **10-user.rules**.

Persistent Names: udevinfo

The default udev rules will provide names for your devices using the official symbolic names reserved for them, for instance, lp*n* for printer, where *n* is the number of the printer. For fixed devices, such as fixed printers, this is normally adequate. However, for removable devices, such as USB printers that may be attached in different sequences at different times to USB ports, the names used may not refer to the same printer. For example, if you have two USB printers, an Epson and Canon, and you attach the Epson first and the Canon second, the Epson will be given the name **usb/lp0** and the Canon will have the name **usb/lp1**. If, however, you later detach them and reattach the Canon first and the Epson second, then the Canon will have the name **usb/lp0** and the Epson will have **usb/lp1**. If you want the Epson to always have the same name, say **epson-pr**, and likewise the Canon, as in **canon-pr**, you will have to create your own rule for detecting these printers and giving them your own symbolic names. The persistent rules are held in the **65-input-persistent.rules** and **65-storage-persistent.rules** files. The key task in creating a persistent name is to use unique information to identify the device. You then create a rule that references the device with the unique information, identifying it, and then name it with an official name but giving it a unique symbolic name. You can then use the unique symbolic name, like **canon-pr**, to always reference just that printer and no other, when it is plugged in. In this example, unique information such as the Canon printer serial number is used to identify the Canon printer. It is next named with the official name, **usb/lp0** or **usb/lp1**, depending on whether another printer was plugged in first, and then it is given a unique symbolic name, **canon-pr**, which will reference that official name, whatever it may be. Keeping the official name, like **lp0**, preserves standard access to the device as used by many applications.

You use **/sys** file system information about the device to detect the correct device to reference with the symbolic link. Unique **/sys** device information such as the vendor serial number or the vendor name can be used to uniquely reference the device. To obtain this information, you need to first query the **/sys** file system. You do this with the **udevinfo** command.

First you will need to know where the device is located in the **/sys** file system. You plug in your device, which will automatically configure and name it, using the official name. For example, plugging in the USB printer will create a **/dev/usb/lp0** device name for it. You can use this device name to find out where the USB printer information is in **/sys**. Use the **udevinfo** command with the **-q path** option to query for the **/sys** pathname, and add the **-n** option with the device's full pathname to identify the device you are searching for. The following command will display the **/sys** path for the printer with the device name **lp0**. In this case, the device is in the **class** subdirectory under **usb**. The path will assume **/sys**.

```
udevinfo -q path -n  /dev/usb/lp0
  /class/usb/lp0
```

One you have the device's **/sys** path, you can use that path to display information about it. Use the **udevinfo** command again with the **-a** option to display all information about the

device and the **-p** option to specify its path in the **/sys** file system. The listing can be extensive, so you should pipe the output to **less** or redirect it to a file.

```
udevinfo -a  -p  /sys/class/usb/lp0 | less
```

Some of the key output to look for is the bus used and information such as the serial number, product name, or manufacturer. Look for information that uniquely identifies the device, such as the serial number. Some devices will support different buses, and the information may be different for each. Be sure to use the information for that bus when setting up your keys in the udev rule.

```
BUS="usb"
ATTRS{serial}="300HCR"
ATTRS{manufacturer}="Canon"
ATTRS{idproduct}="1074"
ATTRS{product}="S330"
```

You can use much of this information in an ATTRS key in an udev rule to identify the device. The ATTRS key (attributes) is used to obtain **/sys** information about a device. You use the ATTRS key with the field you want referenced placed in braces. You can then match that field to a value to reference the particular device you want. Use the **=** sign and a valid field value to match against. Once you know the **/sys** serial number of a device, you can use it in ATTRS keys in udev rules to uniquely reference the device. The following key checks the serial number of the devices field for the Canon printer's serial number:

```
ATTRS{serial}=="300HCR"
```

A user rule can now be created for the Canon printer.

In another rules file you can add your own symbolic link using **/sys** information to uniquely identify the printer and name the device with its official kernel name. The first two keys, BUS and ATTRS, specify the particular printer. In this case the serial number of the printer is used to uniquely identify it. The NAME key will name the printer using the official kernel name, always referenced with the %**k** code. Since this is a USB printer, its device file will be placed in the **usb** subdirectory, **usb/**%**k**. Then the SYMLINK key defines the unique symbolic name to use, in this case **canon-pr** in the **/dev/usb** directory.

```
BUS=="usb", ATTRS{serial}=="300HCR", NAME="usb/%k", SYMLINK="usb/canon-pr"
```

The rules are applied dynamically in real time. To run a new rule, simply attach your USB printer (or detach and reattach). You will see the device files automatically generated.

Permission Fields: MODE, GROUP, OWNER

Permissions that will be given to different device files are determined by the permission fields in the udev rules. On Red Hat, Fedora, and similar distributions the permission rules are located in the **50-udev.rules** file, wheras on Debian and similar distributions they are located in the **40-permissions.rules** file. The MODE field is a octal bit permission setting, the same as is used for file permissions. Usually this is set to 660, owner and group read/write permission. Pattern matching is supported with the *, ?, and [] operators. The following

example sets audio devices to the owner and group with read/write owner and group permission:

```
KERNEL=="audio*",     MODE="0660"
```

The floppy device entry specifies a floppy group.

```
KERNEL=="fd[01]*",     GROUP="floppy", MODE="0660"
```

USB printer devices use the **lp** group with Mode 660.

```
KERNEL=="usb/lp*",     GROUP="lp", MODE="0660"
```

Tape devices use the disk group.

```
KERNEL=="npt*",     GROUP="disk", MODE="0660"
```

The default settings set the OWNER and GROUP to root with owner read/write permissions (600).

```
KERNEL=="*",     OWNER="root" GROUP="root", MODE="0600"
```

Hardware Abstraction Layer: HAL

The purpose of the Hardware Abstraction Layer (HAL) is to abstract the process of applications accessing devices. Applications should not have to know anything about a device, even its symbolic name. The application should just have to request a device of a certain type, and then a service, such as HAL, should provide what is available. Device implementation becomes hidden from applications.

HAL makes devices easily available to desktops and applications using a D-BUS (device bus) structure. Devices are managed as objects that applications can easily access. The D-BUS service is provided by the HAL daemon, **haldaemon**. Interaction with the device object is provided by the **freedesktop.org** HAL service, which is managed by the **/org/freedesktop/HAL/Manager**.

HAL is an information service for devices. The HAL daemon maintains a dynamic database of connected hardware devices. This information can be used by specialized callout programs to maintain certain device configuration files. This is the case with the managing removable storage devices. HAL will invoke the specialized callout programs that will use HAL information to dynamically manage devices. Removable devices like CD-ROM discs or USB card readers are managed by specialized callouts with HAL information, detecting when such items are attached. The situation can be confusing. Callout programs perform the actual tasks, but HAL provides the device information. For example, though the callout hal-system-storage-mount mounts a device, the options and mountpoints used for CD-ROM entries are specified in HAL device information files that set policies for storage management.

HAL has a key impact on the **/etc/fstab** file used to manage file systems. No longer are entries maintained in the **/etc/fstab** file for removable devices like CD-ROMs. These devices are managed directly by HAL using its set of storage callouts like **hal-storage-mount** to

mount a device or **hal-storage-eject** to remove one. In effect you now have to use the HAL device information files to manage your removable file systems.

HAL is a software project of **freedesktop.org**, which specializes in open source desktop tools. Check the latest HAL specification documentation at **freedesktop.org** under the **software/HAL** page for detailed explanations of how HAL works (see the specifications link on the HAL page: Latest HAL Specification). The documentation is very detailed and complete.

The HAL Daemon and hal-device-manager (hal-gnome)

The HAL daemon, **hald**, is run as the **haldaemon** process. Information provided by the HAL daemon for all your devices can be displayed using the HAL device manager. The HAL device manager is part of the hal-gnome package. You can access it, once installed, from the System | Administration | Hardware menu entry. The actual Hal device manager program is named **hal-device-manager**.

When you run the manager, it displays an expandable tree of your devices arranged by category in the left panel. The right panel displays information about the selected device. A Device pane will list the basic device information such as the vendor and the bus type. The Advanced pane will list the HAL device properties defined for this device, as described in later sections, as well as **/sys** file system paths for this device. For device controllers there will also be a USB or PCI panel. For example, a DVD writer could have an entry for the **storage.cdrom.cdr** property that says it can write CD-R discs. You may find an IDE CD/DVD-ROM device under IDE (some third-party IDE controllers may be labeled as SCSI devices). A typical entry would look like this. The **bool** is the type of entry, namely boolean:

```
storage.cdrom.cdr     bool      true
```

Numerical values may use an **int** type or an **strlist** type. The following **write_speed** property has a value 7056:

```
storage.cdrom.write_speed       strlist       7056
```

The **/sys** file system path will also be a string. It will be preceded by a Linux property category. Strings will use a **strlist** type for multiple values and **string** for single values. The following entry locates the **/sys** file system path at **/sys/block/hdc**:

```
linux.sysfs_path     strlist    /sys/block/hdc
```

HAL Configuration: /etc/hal/fdi, and /usr/share/hal/fdi

Information about devices and policies to manage devices are held in device information files in the **/etc/hal/fdi** and **/usr/share/hal/fdi** directories. These directories are where you set properties such as options that are to be used for CD-ROMs in **/etc/fstab**.

The implementation of HAL on Linux configures storage management by focusing on storage methods for mountable volumes, instead of particular devices. Volume properties define actions to take and valid options that can be used. Special callouts perform the actions directly, such as **hal-storage-mount** to mount media, or **hal-storage-eject** to remove it.

Device Information Files: fdi

HAL properties for these devices are handled by device information files (fdi) in the **/usr/share/hal/fdi** and **/etc/hal/fdi** directories. The **/usr/share/hal/fdi** directory is used for configurations provided by the distribution, whereas **/etc/hal/fdi** is used for setting user administrative configurations. In both are listed subdirectories for the different kinds of information that HAL manages, such as **policy**, whose subdirectories have files with policies for how to manage devices. The files, known as device information files, have rules for obtaining information about devices, as well as detecting and assigning options for removable devices. The device information files have the extension **.fdi**, as in **storage-methods.fdi**. For example, the **policy** directory has two subdirectories: **10osvendor** and **20thirdpary**. The **10osvendor** holds the fdi files that have policy rules for managing removable devices on Linux (**10osvendor** replaces **90defaultpolicy** in earlier HAL versions). This directory holds the **20-storage-methods.fdi** policy file used for storage devices. Here you will find the properties that specify options for removable storage devices such as CD-ROMs. The directories begin with numbers; lower numbers are read first. Unlike with udev, the last property read will override any previous property settings, so priority is given to higher-numbered directories and the fdi files they hold. This is why the default policies are in **10osvendor**, whereas the user policies, which override the defaults, are in a higher-numbered directory like **30user**, as are third-party policies, **20thirdpolicy**.

There are currently three device information file directories set up in the device information file directories, each for different kinds of information: information, policy, and preprobe:

- **Information** For information about devices.
- **Policy** For setting policies such as storage policies. The default policies for a storage device are in a **20-storage-methods.fdi** file in the **policy/10osvendor** directory.
- **Preprobe** Handles difficult devices such as unusual drives or drive configurations, for instance, those in **preprobe/10osvendor/10-ide-drives.fdi**. This contains information needed even before the device is probed.

Within these subdirectories are still other subdirectories indicating where the device information files come from, such as **vendor**, **thirdparty**, or **user**, and their priority. Certain critical files are listed here:

- **information/10freedesktop** Information provided by **freedesktop.org**
- **policy/10osvendor** Default policies (set by system administrator and OS distribution)
- **preprobe/10usevendor** Preprobe policies for difficult devices

Properties

Information for a device is specified with a *property* entry. Such entries consist of a key/value pair, where the key specifies the device and its attribute, and the value is the value for that attribute. There are many kinds of values, such as Boolean true/false, string values, such as those used to specify directory mountpoints, or integer values.

Properties are classified according to metadata, physical connection, function, and policies. Metadata provides general information about a device, such as the bus it uses, its driver, or its HAL ID. Metadata properties begin with the key info, as in **info.bus**. Physical properties describe physical connections, namely the buses used. The IDE, PCI, and SCSI bus information is listed in ide, pci, and scsi keys. The usb_device properties are used for the USB bus; an example is **usb_device.number**.

The functional properties apply to specific kinds of devices. Here you will find properties for storage devices, such as the **storage.cdrom** keys that specify if an optical device is writable capabilities. For example, the **storage.cdrom.cdr** key set to true will specify that an optical drive can write to CD-R discs.

The policies are not properties as such. They indicate how devices are to be handled. They are, in effect, the directives that callout programs will use to carry out tasks. Policies for storage media are handled using Volume properties, specifying methods (callouts) to use and valid options, such as mount options. HAL uses scripts in the **/usr/share/hal/scripts** directory to actually manage media. The following abbreviated entries come from the **20-storage-methods.fdi** policy file. The first specifies the action to take and the second the callout script to execute, **hal-storage-mount**.

```
<append key="Volume.method_names" type="strlist">Mount</append>
<append key="Volume.method_execpaths" type="strlist">hal-storage-mount</append>
```

Mount options are designated using **volume.mount.valid_options** as shown here for ro (read only). Options that will be used will be determined when the mount callout is executed.

```
<append key="volume.mount.valid_options"type="strlist">ro</append>
```

Several of the commonly used volume policy properties are listed in Table 31-6.

Property	Description
volume.method.execpath	Callout script to be run for a device
volume.policy.desired_mount_point (string)	The preferred mountpoint for the storage device
volume.mount.valid_options.* (bool)	Mount options to use for specific device, where * can be any mount option, such as **noauto** or **exec**
volume.method_names	Action to be taken
volume.policy.mount_filesystem (string)	File system to use when mounting a volume
volume.mount.valid.mount_ options.* (bool)	Default mount options for volumes, where * can be any mount option, such as **noauto** or **exec**

TABLE 31-6 HAL Storage Policies

Device Information File Directives

Properties are defined in directives listed in device information files. As noted, device information files have **.fdi** extensions. A directive is encased in greater- and less-than symbols. There are three directives:

- **merge** Merges a new property into a device's information database
- **append** Appends or modifies a property for that device already in the database
- **match** Tests device information values

A directive includes a type attribute designating the type of value to be stored, such as string, bool, int, and double. The **copy_property** type copies a property. The following discussion of the **storage-methods.fdi** file shows several examples of merge and match directives.

storage.fdi

The **20-storage-methods.fdi** file in the **/usr/share/hal/fdi/policy/10osvendor** directory lists the policies for your removable storage devices. Here is where your options for storage volumes (for example, CD-ROM) entries are actually specified. The file is organized in sections beginning with particular types of devices to standard defaults. Keys are used to define options, such as **volume.mount.valid_options**, which will specify a mount option for a storage device such as a CD-ROM. Keys are also used to specify exceptions like hotplugged devices.

The **20-storage-methods.fdi** file begins with default properties and then lists those for specific kinds of devices. Unless redefined in a later key, the default will remain in effect. The options you will see listed for the default storage volumes will apply to CD-ROMs. For example, the **noexec** option is set as a valid default. The following sets **noexec** as a default mount option for a storage device. There are also entries for **ro** and **quiet**. The **append** operation adds the policy option.

```
<append key="volume.mount.valid_options"type="strlist">noexec</append>
```

The default mountpoint root directory for storage devices is now set by the mount callout script, **hal-storage-mount**. Currently this is **/media**. The default mountpoint is disk. HAL will try to use the Volume property information to generate a mountpoint.

The following example manages blank disks. Instead of being mounted, such disks can only be ejected. To determine possible actions, HAL uses **method_names**, **method_signatures**, and **method_execpaths** for the Volume properties (the **org.freedesktop.Hal** prefix for the keys has been removed from this example to make it more readable, as in **org.freedesktop.Hal.Volume.method_names**).

```
<match key="volume.disc.is_blank" bool="true">
<append key="info.interfaces"type="strlist">Volume</append>
<append key="Volume.method_names" type="strlist">Eject</append>
<append key="Volume.method_signatures" type="strlist">as</append>
<append key="Device.Volume.method_execpaths" type="strlist">hal-storage-eject</append>
</match>
```

After dealing with special cases, the file system devices are matched as shown here:

```
<match key="volume.fsusage" string="filesystem">
```

Storage devices to ignore, such as vfat, are specified.

```
<merge key="volume.ignore" type="bool">false</merge>
```

Then the actions to take and the callout script to use are specified, such as the one for Unmount that uses **hal-storage-mount**.

```
<append key="Device.Volume.method_names" type="strlist">Mount</append>
<append key="Device.Volume.method_signatures" type="strlist">ssas</append>
<append key="Device.Volume.method_execpaths" type="strlist">hal-storage-mount</append>
```

Options are then specified with **volume.mount.valid_options**, starting with defaults and continuing with special cases, such as **ext3** shown here.

```
<!-- allow these mount options for ext3 -->
<match key="volume.fstype" string="ext3">
<append key="volume.mount.valid_options"type="strlist">data=</append>
 </match>
```

HAL Callouts

Callouts are programs invoked when the device object list is modified or when a device changes. As such, callouts can be used to maintain systemwide policy (that may be specific to the particular OS) such as changing permissions on device nodes, managing removable devices, or configuring the networking subsystem. There are three different kinds of callouts for devices, capabilities, and properties. *Device* callouts are run when a device is added or removed. *Capability* callouts add or remove device capabilities, and *property* callouts add or remove a device's property. In the current release, callouts are implemented using info.callout property rules, such as which invokes the **hal-storage-mount** callout when CD/DVD-ROMs are inserted or removed as shown here:

```
<append key="org.freedesktop.Hal.Device.Volume.method_execpaths"
type="strlist">hal-storage-mount</append>
```

Callouts are placed in the **/usr/libexec** directory with the HAL callouts prefixed with **hal-**. Here you will find many storage callouts used by HAL such as **hal-storage-eject** and **hal-storage-mount**. HAL uses these callouts to manage removable devices like DVD/CD-ROMs directly instead of editing entries in the **/etc/fstab** file (**fstab-sync** is no longer used). The **gnome-mount** tool used for mounting CD/DVD disk on the GNOME desktop uses the HAL callouts. Other supporting scripts can be found in the **/usr/lib/hal/scripts** directory.

Manual Devices

Several devices still need to be created manually: printer parallel ports, for example. Most of these devices are already configured with **MAKEDEV** and the **/etc/makedev.d** files. To have these devices created by udev, their names are placed in configuration files in the **/etc/udev/makedev.d** directory. The **50-udev.nodes** file contains a list of device names that udev will

use **MAKEDEV** to manually construct when udev generates the **/dev** device directory. Here you will find entries for parallel ports like **paraport0** through **parport3**.

You can, if you wish, create device file interfaces manually yourself using the **MAKEDEV** or **mknod** commands. To have them added to the **/dev** directory by udev, place them in the **/etc/udev/devices** directory; udev will copy them for you to the **/dev** directory when it generates them. For some devices, such as ISDN devices, you may have to do this. The following example makes an ISDN device using **MAKEDEV** and places it in the **/etc/udev/devices** directory.

```
/sbin/MAKEDEV -d /etc/udev/devices isdn
```

Device Types

Linux implements several types of devices, the most common of which are block and character. A *block device,* such as a hard disk, transmits data a block at a time. A *character device,* such as a printer or modem, transmits data one character at a time, or rather as a continuous stream of data, not as separate blocks. Device driver files for character devices have a *c* as the first character in the permissions segment displayed by the **ls** command. Device driver files for block devices have a *b.* In the next example, **lp0** (the printer) is a character device and **sda1** (the hard disk) is a block device:

```
# ls -l sda1 lp0
brw-rw---- 1 root disk 3, 1 Jan 30 02:04 sda1
crw-rw---- 1 root lp   6, 0 Jan 30 02:04 lp0
```

The device type can be either *b, c, p,* or *u.* As already mentioned, the *b* indicates a block device, and *c* is for a character device. The *u* is for an unbuffered character device, and the *p* is for a FIFO (first in, first out) device. Devices of the same type often have the same name; for example, serial interfaces all have the name **ttyS**. Devices of the same type are then uniquely identified by a number attached to the name. This number has two components: the major number and the minor number. Devices may have the same major number, but if so, the minor number is always different. This major and minor structure is designed to deal with situations in which several devices may be dependent on one larger device, such as several modems connected to the same I/O card. All the modems will have the same major number that references the card, but each modem will have a unique minor number. Both the minor and major numbers are required for block and character devices (*b, c,* and *u*). They are not used for FIFO devices, however.

Valid device names along with their major and minor numbers are listed in the **devices.txt** file located in the **/Documentation** directory for the kernel source code, **/usr/src/linux-***ver***/Documentation**. When you create a device, you use the major and minor numbers as well as the device name prefix for the particular kind of device you are creating. Most of these devices are already created for you and are listed in the **/etc/dev** directory.

MAKEDEV

You use **MAKEDEV** to create device files. **MAKEDEV** uses device configuration files located in the **/etc/makedev.d** directory to determine device options like the major or minor number of the device or any symbolic links that should be created for it. For example, the

/etc/makedev.d/01sound file lists sound devices. A **MAKEDEV** configuration file can have three different kinds of records, each beginning with a different operator:

- **b or c** Creates a block (b) or character (c) device. These entries hold several options: mode (permissions), owner, group, major and minor numbers, inc, count (number of devices created), and fmt (the name of the device). The fmt option is technically a format string, which can include a format specifier for numerically incrementing names of the similar devices, such as **cdrom%d** for **cdrom0**, **cdrom1**, and so on. The inc option sets an increment.
- **l** Creates a symbolic link for a device.
- **a** An alias that applies the commands used for one device for those of another. This lets you create a sound device, which in turn automatically creates audio, MIDI, and mixer devices, and so on.

In the **/etc/makedev.d/01sound** file, there are numerous alias entries for sound, such as the following:

```
a  sound audio
```

A link entry will create a symbolic link called **audio0** for the audio device file.

```
l audio0 audio
```

The actual sound device file creation is configured in the **alsa** file. Sound devices use ALSA sound drivers. Here you will find numerous c entries with permission, owner, group values, etc.

With so much of the configuration handled in the **MAKEDEV** device configuration files, the command to create a device is very simple. However, bear in mind that with udev, device files cannot be created in **/dev**. This directory is automatically regenerated by udev. To have udev place your device file in **/dev** when it generates it, you place the device file you made in **/etc/udev/devices**. Use the **-d** option to specify the udev device directory. The following creates an ISDN device:

```
MAKEDEV -d /etc/udev/devices  isdn
```

mknod

Though the **MAKEDEV** command is preferable for creating device files, it can only create files for which it is configured. For devices not configured for use by **MAKEDEV**, you will have to use the **mknod** command. This is a lower-level command that requires manual configuration of all its settings. With the **mknod** command you can create a device file in the traditional manner without any of the configuration support that **MAKEDEV** provides.

The **mknod** command can create either a character or block-type device. The **mknod** command has the following syntax:

```
mknod options device device-type major-num minor-num
```

As most devices are easily covered by **MAKEDEV** as well as automatically generated by udev, you will rarely if ever need to use **mknod**. As a simple example, creating a device file

with **mknod** for a printer port is discussed here. Linux systems usually provide device files for printer ports (**lp0–2**). As an example, you can see how an additional port could be created manually with the **mknod** command. Printer devices are character devices and must be owned by the root and daemon. The permissions for printer devices are read and write for the owner and the group, 660. The major device number is set to 6, while the minor device number is set to the port number of the printer, such as 0 for LPT1 and 1 for LPT2. Once the device is created, you use **chown** to change its ownership to the **root** user, since only the administrator should control it. Change the group to **lp** with the **chgrp** command.

Most devices belong to their own groups, such as **disks** for hard disk partitions, **lp** for printers, **floppy** for floppy disks, and **tty** for terminals. In the next example, a printer device is made on a fourth parallel port, **lp3**. The **-m** option specifies the permissions—in this case, 660. The device is a character device, as indicated by the **c** argument following the device name. The major number is 6, and the minor number is 3. If you were making a device at **lp4**, the major number would still be 6, but the minor number would be 4. Once the device is made, the **chown** command then changes the ownership of the parallel printer device to **root**. For printers, be sure that a spool directory has been created for your device. If not, you need to make one. Spool directories contain files for data that varies according to the device output or input, like that for printers or scanners.

As with all manual devices, the device file has to be placed in the **/etc/udev/devices** directory; udev will later put it in **/dev**.

```
# mknod -m 660 /etc/udev/devices/lp3 c 6 3
# chown root /etc/udev/devices/lp3
# chgrp lp /etc/udev/devices/lp3
```

Installing and Managing Terminals and Modems

In Linux, several users may be logged in at the same time. Each user needs his or her own terminal through which to access the Linux system, of course. The monitor on your PC acts as a special terminal, called the *console,* but you can add other terminals through either the serial ports on your PC or a special multiport card installed on your PC. The other terminals can be stand-alone terminals or PCs using terminal emulation programs. For a detailed explanation of terminal installation, see the **Term-HOWTO** file in **/usr/share/doc/HOWTO** or at the Linux Documentation Project site (**tldp.org**). A brief explanation is provided here.

Serial Ports

The serial ports on your PC are referred to as COM1, COM2, COM3, and COM4. These serial ports correspond to the terminal devices **/dev/ttyS0** through **/dev/ttyS3**. Note that several of these serial devices may already be used for other input devices such as your mouse and for communications devices such as your modem. If you have a serial printer, one of these serial devices is already used for that. If you installed a multiport card, you have many more ports from which to choose. For each terminal you add, udev will create the appropriate character device on your Linux system. The permissions for a terminal device are normally 660. *Terminal devices* are character devices with a major number of 4 and minor numbers usually beginning at 64.

TIP *The **/dev/pts** entry in the **/etc/fstab** file mounts a **devpts** file system at **/dev/pts** for Unix98 Pseudo-TTYs. These pseudoterminals are identified by devices named by number.*

mingetty, mgetty, and agetty

Terminal devices are managed by your system using the **getty** program and a set of configuration files. When your system starts, it reads a list of connected terminals in the **inittab** file and then executes an appropriate **getty** program for each one, either **mingetty**, **mgetty**, or **agetty**. Such **getty** programs set up the communication between your Linux system and a specified terminal. **mingetty** provides minimal support for virtual consoles, whereas **agetty** provides enhanced support for terminal connections. **agetty** also includes parameters for the baud rate and timeout. **mgetty** is designed for fax/modem connections, letting you configure dialing, login, and fax parameters. **mgetty** configuration files are held in the **/etc/mgetty+sendfax** directory. Modem connection information is held in the **/etc/mgetty+sendfax/mgetty.config** file. All **getty** programs can read an initial message placed in the **/etc/issue** file, which can contain special codes to provide the system name and current date and time.

termcap and inittab Files

The **/etc/inittab** file holds instructions for your system on how to manage terminal devices. A line in the **/etc/inittab** file has four basic components: an ID, a runlevel, an action, and a process. Terminal devices are identified by ID numbers, beginning with 1 for the first device. The runlevel at which the terminal operates is usually 1. The action is usually *respawn*, which means to run the process continually. The process is a call to the **mingetty**, **mgetty**, or **agetty** with the terminal device name. The **/etc/termcap** file holds the specifications for different terminal types. These are the different types of terminals users can use to log in to your system. Your **/etc/termcap** file is already filled with specifications for most of the terminals currently produced. An entry in the **/etc/termcap** file consists of various names that can be used for a terminal separated by a pipe character (|) and then a series of parameter specifications, each ending in a colon. You find the name used for a specific terminal type here. You can use **more** to display your **/etc/termcap** file, and then use a search, **/**, to locate your terminal type. You can set many options for a terminal device. To change these options, use the **stty** command instead of changing configuration files directly. The **stty** command with no arguments lists the current setting of the terminal.

tset

When a user logs in, having the terminal device initialized using the **tset** command is helpful. Usually the **tset** command is placed in the user's **.bash_profile** file and is automatically executed whenever the user logs in to the system. You use the **tset** command to set the terminal type and any other options the terminal device requires. A common entry of **tset** for a **.bash_profile** file follows. The **-m dialup:** option prompts the user to enter a terminal type. The type specified here is a default type that is displayed in parentheses. The user presses ENTER to choose the default. The prompt looks like this: **TERM=(vt100)?**.

```
eval 'tset -s -Q -m dialup:?vt00'
```

Input Devices

Input devices, such as mice and keyboards, are displayed on several levels. Initial detection is performed during installation where you select the mouse and keyboard types. Keyboard and mice will automatically be detected by HAL. You can perform detailed configuration with your desktop configuration tools, such as the GNOME or KDE mouse configuration tools. On GNOME, select System | Preferences | Hardware | Mouse to configure your mouse. There is a Keyboard entry on that same menu for keyboards.

Installing Sound, Network, and Other Cards

For you to install a new card, your kernel must first be configured to support it. Support for most cards is provided in the form of modules that can be dynamically loaded into the kernel. Installing support for a card is usually a simple matter of loading a module that includes the drivers for it. For example, drivers for the Sound Blaster sound card are in the module **sb.o**. Loading this module makes your sound card accessible to Linux. Most Linux distributions automatically detect the cards installed on your system and load the needed modules. If you change sound cards, the new card is automatically detected. You can also load modules you need manually, removing an older conflicting one. The section "Modules" later in this chapter describes this process.

Device files for most cards are already set up for you in the **/dev** directory by **udev**. For example, the device name for your sound card is **/dev/audio**. However, the device names for network cards are aliases for network modules instead of device files. For example, the device name for your Ethernet card begins with **eth,** with the numbering starting from **0**, as in **eth0** for the first Ethernet card on your system. An alias used to reference the module used for that particular card; for example, a 3Com Etherlink card aliases the 3c59x network module, whose alias would be **eth0** if it is the first Ethernet card. The modules themselves are kept in the kernel's module directory located at **/lib/modules**, as described in the section "Module Files and Directories: /lib/modules."

Sound Devices

Your Linux distribution will usually provide a configuration tool to configure most sound cards. Most sound cards are now detected and managed by udev and HAL. A listing of the different sound devices is provided in Table 31-7. Some sound cards may require more

Device	Description
/dev/sndstat	Sound driver status
/dev/audio	Audio output device
/dev/dsp	Sound sampling device
/dev/mixer	Control mixer on sound card
/dev/music	High-level sequencer
/dev/sequencer	Low-level sequencer
/dev/midi	Direct MIDI port

TABLE 31-7 Sound Devices

specialized support. For sound cards, you can tell what your current sound configuration is by listing the contents of the **/proc/asound/oss/sndstat** file. You can test your card by simply redirecting a sound file to it, as shown here:

```
cat sample.au > /dev/audio
```

For the 2.4 kernel, most Linux sound drivers were developed as part of the Open Sound System (OSS) and freely distributed as OSS/Free. These are installed as part of Linux distributions. The OSS device drivers are intended to provide a uniform API for all Unix platforms, including Linux. They support Sound Blaster– and Windows Sound System–compatible sound cards (ISA and PCI). OSS is also available for a nominal fee and features configuration interfaces for device setup.

The Advanced Linux Sound Architecture (ALSA) replaced OSS in the 2.6 Linux kernel; it aims to be a better alternative to OSS, while maintaining compatibility with it. ALSA provides a modular sound driver, an API, and a configuration manager. ALSA is a GNU project and is entirely free; its website at **alsa-project.org** contains extensive documentation, applications, and drivers. Currently available are the ALSA sound driver, the ALSA Kernel API, the ALSA library to support application development, and the ALSA manager to provide a configuration interface for the driver. ALSA evolved from the Linux Ultra Sound Project. The ALSA project currently supports most Creative sound cards.

The Linux Musical Instrument Digital Interface (MIDI) and Sound Pages, currently at **linux-sound.org**, includes links to sites for Linux MIDI and sound software.

Video and TV Devices

Device names used for TV, video, and DVD devices are listed in Table 31-8. Drivers for DVD and TV decoders have been developed, and mga4linux (**marvel.sourceforge.net**) is developing video support for the Matrox Multimedia cards. The General ATI TV and Overlay Software (GATOS) (**gatos.sourceforge.net**) has developed drivers for the currently unsupported features of ATI video cards, specifically TV features. The BTTV Driver Project has developed drivers for the Booktree video chip. Creative Labs sponsors Linux drivers for the Creative line of DVD DXR2 decoders (**opensource.creative.com**).

Device Name	Type of Device
/dev/video	Video capture interface
/dev/vfx	Video effects interface
/dev/codec	Video codec interface
/dev/vout	Video output interface
/dev/radio	AM/FM radio devices
/dev/vtx	Teletext interface chips
/dev/vbi	Data services interface

TABLE 31-8 Video and TV Device Drivers

PCMCIA Devices

PCMCIA devices are card readers commonly found on laptops to connect devices such as modems or wireless cards, though they are becoming standard on many desktop systems as well. The same PCMCIA device can support many different kinds of devices, including network cards, modems, hard disks, and Bluetooth devices.

PCMCIA support is now managed by udev and HAL; you can no longer use the cardmgr/pcmcia service. PCMCIA devices are now considered hotplugged devices managed by HAL and udev directly. Card information and control is now managed by **pccardctl**. The PCMCIA udev rules are listed in **60-pcmcia.rules**, which automatically probes and installs cards. Check **kernel.org/pub/linux/utils/kernel/pcmcia/pcmcia.html** for more information.

You can obtain information about a PCMCIA device with the **pccardctl** command, as well as manually eject and insert a device. The **status**, **config**, and **ident** options will display the device's socket status and configuration and the identification of the device. The **insert** and **eject** options will let you add and remove a device. The **cardinfo** command also provides device information.

It is not advisable to hot-swap IDE or SCSI devices. For these you should first manually shut down the device using the **pccardctl** command.

```
pccardctl eject
pccardctl scheme home
```

Modules

The Linux kernel employs the use of modules to support different operating system features, including support for various devices such as sound and network cards. In many cases, you do have the option of implementing support for a device either as a module or by directly compiling it as a built-in kernel feature, which requires you to rebuild the kernel. A safer and more robust solution is to use modules. *Modules* are components of the Linux kernel that can be loaded as needed. To add support for a new device, you can now simply instruct a kernel to load the module for that device. In some cases, you may have to recompile only that module to provide support for your device. The use of modules has the added advantage of reducing the size of the kernel program as well as making your system more stable. The kernel can load modules in memory only as they are needed. Should a module fail, only the module stops running, and it will not affect the entire system. For example, the module for the PPP network interface used for a modem needs to be used only when you connect to an ISP.

Kernel Module Tools

The modules your system needs are usually determined during installation, according to the kind of configuration information you provided and the automatic detection performed by your Linux distribution. For example, if your system uses an Ethernet card whose type you specified during installation, the system loads the module for that card. You can, however, manually control what modules are to be loaded for your system. In effect, this enables you to customize your kernel whatever way you want. You can use several commands, configuration tools, and daemons to manage kernel modules. The 2.6 Linux kernel includes the Kernel Module Loader (Kmod), which has the capability to load modules automatically

as they are needed. Kernel module loading support must also be enabled in the kernel, though this is usually considered part of a standard configuration. In addition, several tools enable you to load and unload modules manually, if you must. The Kernel Module Loader uses certain kernel commands to perform the task of loading or unloading modules. The **modprobe** command is a general-purpose command that calls **insmod** to load modules and **rmmod** to unload them. These commands are listed in Table 31-9. Options for particular modules, general configuration, and even specific module loading can specified in the **/etc/modprobe.conf** file. You can use this file to automatically load and configure modules. You can also specify modules to be loaded at the boot prompt or in **grub.conf**.

Module Files and Directories: /lib/modules

The filename for a module has the extension **.o**. Kernel modules reside in the **/lib/modules/***version* directory, where *version* is the version number for your current kernel with the extension FC7. The directory for the 2.6.20-1.2054_FC7 kernel is **/lib/modules/2.6.20-1.2054_FC5**. As you install new kernels on your system, new module directories are generated for them. One method to access the directory for the current kernel is to use the **uname -r** command to generate the kernel version number. This command needs to have back quotes.

```
cd /lib/modules/`uname -r`
```

In this directory, modules for the kernel reside in the **/kernel** directory. Within the **/kernel** directory are several subdirectories, including the **/drivers** directory that holds subdirectories for modules like sound drivers or video drivers. These subdirectories serve to categorize your modules, making them easier to locate. For example, the **kernel/drivers/net** directory holds modules for your Ethernet cards, and the **kernel/drivers/sound** directory contains sound card modules.

TIP *You will notice that there are no entries for the Ethernet devices in the **/dev** file, such as **eth0** or **eth1**. That is because these are really aliases for kernel modules defined in the **/etc/modprobe.conf** file, or devices handled by the kernel directly. They are not device files.*

Command	Description
lsmod	Lists modules currently loaded.
insmod	Loads a module into the kernel. Does not check for dependencies.
rmmod	Unloads a module currently loaded. Does not check for dependencies.
modinfo	Displays information about a module: **-a** (author), **-d** (description), **-p** (module parameters), **-f** (module filename), **-v** (module version).
depmod	Creates a dependency file listing all other modules on which the specified module may rely.
modprobe	Loads a module with any dependent modules it may also need. Uses the file of dependency listings generated by **depmod**: **-r** (unload a module), **-l** (list modules).

TABLE 31-9 Kernel Module Commands

Managing Modules with modprobe

As noted previously, there are several commands you can use to manage modules. The **lsmod** command lists the modules currently loaded into your kernel, and **modinfo** provides information about particular modules. Though you can use the **insmod** and **rmmod** commands to load or unload modules directly, you should use only **modprobe** for these tasks. Often, however, a given module requires other modules to be loaded. For example, the module for the Sound Blaster sound card, **sb.o**, requires the **sound.o** module to be loaded also.

The depmod Command

Instead of manually trying to determine what modules a given module depends on, you use the **depmod** command to detect the dependencies for you. The **depmod** command generates a file that lists all the modules on which a given module depends. The **depmod** command generates a hierarchical listing, noting what modules should be loaded first and in what order. Then, to load the module, you use the **modprobe** command using that file. **modprobe** reads the file generated by **depmod** and loads any dependent modules in the correct order, along with the module you want. You need to execute **depmod** with the **-a** option once, before you ever use **modprobe**. Entering **depmod -a** creates a complete listing of all module dependencies. This command creates a file called **modules.dep** in the module directory for your current kernel version, **/lib/modules/***version*.

```
depmod -a
```

The modprobe Command

To install a module manually, you use the **modprobe** command and the module name. You can add any parameters the module may require. The following command installs the Intel high definition sound module. **modprobe** also supports the use of the * character to enable you to use a pattern to select several modules. This example uses several values commonly used for sound cards. You use the values recommended for your sound card on your system. Most sound card drivers are supported by the ALSA project. Check their website to find what driver module is used for your card.

```
modprobe  snd-hda-intel
```

To discover what parameters a module takes, you can use the **modinfo** command with the **-p** option.

You can use the **-l** option to list modules and the **-t** option to look for modules in a specified subdirectory. Sound modules are arranged in different subdirectories according to the device interface they use, like **pci**, **isa**, or **usb**. Most internal sound cards use **pci**. Within the interface directory, there may be further directories like **emu10k1** used for the Creative Audigy cards and **hda** for high definition drivers. In the next example, the user lists all modules in the **sound/pci/hda** directory:

```
# modprobe -l -t sound/pci/hda
/lib/modules/2.6.15-1.3059_FC7/kernel/sound//pci/hda/snd-hda-intel.o
/lib/modules/2.6.15-1.3059_FC7/kernel/sound//pci/hda/snd-hda-codec.o
/lib/modules/2.6.15-1.2054_FC5/kernel/drivers/sound/sound.o
/lib/modules/2.6.15-1.2054_FC5/kernel/drivers/sound/soundcore.o
```

Options for the **modprobe** command are placed in the **/etc/modprobe.conf** file. Here, you can enter configuration options, such as default directories and aliases. An alias provides a simple name for a module. For example, the following entry enables you to reference the **3c59x.o** Ethernet card module as **eth0** (Kmod automatically detects the 3Com Ethernet card and loads the 3c59x module):

```
alias eth0 3c59x
```

On Nvidia systems, the forcedeth module is used.

```
alias eth0 forcedeth
```

The insmod Command

The **insmod** command performs the actual loading of modules. Both **modprobe** and the Kernel Module Loader make use of this command to load modules. Though **modprobe** is preferred because it checks for dependencies, you can load or unload particular modules individually with **insmod** and **rmmod** commands. The **insmod** command takes as its argument the name of the module, as does **rmmod**. The name can be the simple base name, like **sb** for the **sb.o** module. You can specify the complete module filename using the **-o** option. Other helpful options are the **-p** option, which lets you probe your system first to see if the module can be successfully loaded, and the **-n** option, which performs all tasks except actually loading the module (a dummy run). The **-v** option (verbose) lists all actions taken as they occur. In those rare cases where you may have to force a module to load, you can use the **-f** option. In the next example, **insmod** loads the **sb.o** module:

```
# insmod -v sb
```

The rmmod Command

The **rmmod** command performs the actual unloading of modules. It is the command used by **modprobe** and the Kernel Module Loader to unload modules. You can use the **rmmod** command to remove a particular module as long as it is not being used or required by other modules. You can remove a module and all its dependent modules by using the **-r** option. The **-a** option removes all unused modules. With the **-e** option, when **rmmod** unloads a module, it saves any persistent data (parameters) in the persistent data directory, usually **/var/lib/modules/persist**.

modprobe Configuration

Module loading can require system renaming, as well as specifying options to use when loading specific modules. Even when removing or installing a module, certain additional programs may have to be run or other options specified. These parameters can be set in a **/etc/modprobe.conf** file or in files located in an **/etc/modprobe.d** directory. Configuration for modprobe supports four actions: alias, options, install, and remove.

- **alias** *module name* Provides another name for the module, used for network and sound devices.
- **options** *module options* Specifies any options a particular module may need.
- **install** *module commands* Uses the specified commands to install a module, letting you control module loading.

- **remove** *module commands* Specifies commands to be run when a module is unloaded.
- **include** *config-file* Additional configuration files.
- **blacklist** *module* Ignore any internal aliases that given module may define for itself. This allows you to use only aliases defined by modprobe. Also avoids conflicting modules where two different modules may have the same alias defined interally. Default blacklist entries are held in one or more blacklist files in the **/etd/modprobe.d** directory. Their names begin with the term **blacklist**. Use the **modinfo** command to list a module's internal aliases.

Among the more common entries are aliases used for network cards. Notice that there is no device name for Ethernet devices in the **/dev** directory. This is because the device name is really an alias for an Ethernet network module that has been defined in a modprobe configuration file (this was called **modules.conf** in previous releases). To add another Ethernet card of the same type, you place an alias for it in the modprobe configuration file. For a second Ethernet card, you use the device name **eth1** as the alias. This way, the second Ethernet device can be referenced with the name **eth1**. A modprobe configuration entry is shown here:

```
alias eth1 ne2k-pci
```

TIP *After making changes to **/etc/modprobe.conf** or **modprobe.d** files, you should run **depmod** again to record any changes in module dependencies.*

If you add a different model Ethernet card, you have to specify the module used for that kind of card. In the following example, the second card is a standard PCI Realtek card. Kmod has already automatically detected the new card and loaded the **ne2k-pci** module for you. You only need to identify this as the **eth1** card in a modprobe configuration file like **/etc/ modprobe.conf**.

```
alias eth0 forcedeth
alias eth1 ne2k-pci
```

NOTE *Instead of a single **modprobe.conf** file, modprobe configuration can be implemented using separate files in an **/etc/modprobe.d** directory.*

A sample **modprobe.conf** file is shown here. Notice the aliases for the Nvidia Serial ATA controller, **sata_nv**, and the AMD parallel ATA adapter, **pata_amd**. Both are aliased as SCSI host adapters.The sound card is referenced by its module name, **snd-hda-intel**, in later install and remove operations. The last two lines are one line, beginning with **remove**.

```
alias eth0 forcedeth
alias eth1 ne2k-pci
alias scsi_hostadapter sata_nv
alias scsi_hostadapter1 pata_amd
alias snd-card-0 snd-hda-intel
options snd-card-0 index=0
options snd-hda-intel index=0
```

TIP *In some cases, Kmod may not detect a device in the way you want and thereby not load the kernel module you would like. In this case, kernel parameters were specified to the GRUB boot loader to load the correct modules.*

Installing New Modules from Vendors: Driver Packages

Often you may find that your hardware device is not supported by current Linux modules. In this case you may have to download drivers from the hardware vendor or open source development group to create your own driver and install it for use by your kernel.

The drivers could be in RPM or compressed archives. The process for installing drivers differs, depending on how a vendor supports the driver. Different kinds of packages are listed here:

- **RPM or Deb packages** Some support sites will provide drivers already packaged in RPM or Deb files for direct installation.
- **Drivers compiled in archives** Some will provide drivers already compiled for your distribution but packaged in compressed archives. In this case a simple install operation will place the supporting module in the **modules** directory and make if available for use by the kernel.
- **Source code** Others provide just the source code, which, when compiled, will detect your system configuration and compile the module accordingly.
- **Scripts with source code** Some will provide customized scripts that may prompt you for basic questions about your system and then both compile and install the module.

For drivers that come in the form of compressed archives (**tar.gz** or **tar.bz2**), the compile and install operations normally use a Makefile script operated by the **make** command. A simple install usually just requires running the following command in the driver's software directory:

```
make install
```

In the case of sites that supply only the source code, you may have to perform both configure and compile operations as you would for any software.

```
./configure
make
make install
```

For packages that have no install option, compiled or source, you will have to manually move the module to the kernel module directory, **/lib/modules**/*version*, and use **depmod** and **modprobe** to load it (see the preceding section).

If a site gives you a customized script, you just run that script. For example, the Marvel gigabit LAN network interfaces found on many motherboards use the SysKonnect Linux drivers held in the sk98lin.o module. The standard kernel configuration will generate and install this module. But if you are using a newer motherboard, you may need to download and install the latest Linux driver. For example, some vendors may provide a script, **install.sh**, that you run to configure, compile, and install the module.

```
./install.sh
```

NOTE *On Red Hat and Fedora, if you are provided with only a source code file for the module, such as a* ***.c*** *file, you can use the kernel header files in the* ***/lib/modules/***version***/build*** *directory to compile the module. See the Fedora or Red Hat Release Note for details on how to create a customized Makefile for creating modules. You will not have to download and install the source code.*

Installing New Modules from the Kernel

The source code for your Linux kernel contains an extensive set of modules, all of which are not actually used on your system. The kernel binaries provided by most distributions come with an extensive set of modules already installed. If, however, you install a device for which kernel support is not already installed, you will have to configure and compile the kernel module that provides the drivers for it. This involves using the kernel source code to select the module you need from a list in a kernel configuration tool and then regenerating your kernel modules with the new module included (see Chapter 32). Then the new module is copied into the module library, installing it on your system. You can also enter it in the **/etc/modprobe.conf** file with any options, or use **`modprobe`** to install it manually.

Download the original source code version of the kernel in the compressed archive from **kernel.org**, then unpack it in any directory (but do not use **/usr/src/linux**). Alternatively, you can use a prepackaged version of the kernel source provided by your distribution.

Now change to the kernel directory and use the **`make`** command with the **`gconfig`** or **`menuconfig`** argument to display the kernel configuration menus, invoking them with the following commands. The **`make gconfig`** command starts an X Window System interface that needs to be run on your desktop from a terminal window.

```
make gconfig
```

Using the menus, select the modules you need. Make sure each is marked as a module, clicking the Module check box in **`gconfig`** or typing **`m`** for **`menuconfig`**. Once the kernel is configured, save it and exit from the configuration menus. Then you compile the modules, creating the module binary files with the following command:

```
make modules
```

This places the modules in the kernel source modules directory. You can copy the one you want to the kernel modules directory, **/lib/modules/***version***/kernel**, where *version* is the version number of your Linux kernel. A simpler approach is to reinstall all your modules, using the following command. This copies all the compiled modules to the **/lib/modules/***version***/kernel** directory:

```
make modules_install
```

NOTE *If you are using Red Hat or Fedora, first make sure you have installed the kernel source code in the* ***/usr/src/redhat/BUILD*** *directory.*

CHAPTER 32

Kernel Administration

The *kernel* is the operating system, performing core tasks such as managing memory and disk access, as well as interfacing with the hardware that makes up your system. For example, the kernel makes possible such standard Linux features as multitasking and multiuser support. It also handles communications with devices like your CD-ROM or hard disk. Users send requests for access to these devices through the kernel, which then handles the lower-level task of actually sending appropriate instructions to a device. Given the great variety of devices available, the kind of devices connected to a Linux system will vary. When Linux is installed, the kernel is appropriately configured for your connected devices. However, if you add a new device, you may have to enable support for it in the kernel. This involves reconfiguring the existing kernel to support the new device through a procedure that is often referred to as *building* or *compiling the kernel.* In addition, new versions of the kernel are continuously made available that provide improved support for your devices, as well as support for new features and increased reliability for a smoother-running system. You can download, compile, and install these new versions on your system.

Kernel Versions

The version number for a Linux kernel consists of four segments: the major, minor, revision, and security/bug fix numbers. The *major number* increments with major changes in the kernel and is rarely changed. The *minor number* indicates a major revision of the kernel. The revision number is used for supporting new features. The security/bug number is used for security and bug fixes. New development versions will first appear as release candidates, which will have an **rc** in the name. As bugs are discovered and corrected, and as new features are introduced, new revisions of a kernel are released. For example, kernel 2.6.21.1 has a major number of 2 and a minor number of 6, with a revision number of 21, and a security/bug fix number of 1. A release candidate version for a new kernel will have a name like **2.6.22-rc3**.

Distributions often add another number that refers to a specific set of patches applied to the kernel, as well as a distribution initial. For example, the kernel is 2.6.21-1.3116, where **3116** is the patch number. On distributions that support RPM packages, you can use an RPM query to learn what version is installed, as shown here:

```
rpm -q kernel
```

You can have more than one version of the kernel installed on your system. To see which one is running currently, you use the **uname** command with the **-r** option (the **-a** option provides more detailed information).

```
uname -r
```

The Linux kernel is being worked on constantly, and new versions are released when they are ready. Distributions may include different kernel versions. Linux kernels are available at **kernel.org**. Also, RPM packages for a new kernel are often available at distribution update sites. One reason you may need to upgrade your kernel is to provide support for new hardware or for features not supported by your distribution's version. For example, you may need support for a new device not provided in your distribution's version of the kernel. Certain features may not be included in a distribution's version because they are considered experimental or a security risk.

NOTE *In many cases, you don't need to compile and install a new kernel just to add support for a new device. Kernels provide most device support in the form of loadable modules, of which only those needed are installed with the kernel. Most likely, your current kernel has the module you need; you simply have to compile it and install it.*

TIP *Many modules can be separately compiled using sources provided by vendors, such as updated network device drivers. For these you only need the Kernel headers, which are already installed in the* ***/usr/lib/modules/***version***/build*** *directory, where version is an installed kernel version. In these cases, you do not have to install the full kernel source to add or modify modules.*

References

You can learn more about the Linux kernel from **kernel.org**, the official repository for the current Linux kernels. The most current source code, as well as documentation, is there. Your distribution website will also provide online documentation for installing and compiling the kernel on its systems. Several Linux HOWTOs also exist on the subject. The kernel source code software packages also include extensive documentation. Kernel source code files are always installed either directly in a local directory or in the directory used by distribution kernel packages. The source itself will be in a directory labeled **linux-***version*, where *version* is the kernel release, as in **linux-2.6.21**. In this directory, you can find a subdirectory named **/Documentation,** which contains an extensive set of files and directories, documenting kernel features, modules, and commands. The following listing of kernel resources also contains more information:

- **kernel.org** The official Linux kernel website; all new kernels originate from here
- **linuxhq.com** Linux headquarters, kernel sources, and patches
- **kernelnewbies.org** Linux kernel sources and information
- **en.tldp.org** Linux Documentation Project

Kernel Tuning: Kernel Runtime Parameters

Several kernel features, such as IP forwarding or the maximum number of files, can be turned on or off without compiling and installing a new kernel or module. These tunable parameters are controlled by the files in **/proc/sys** directory. Parameters that you set are made in the **/etc/sysctl.conf** file. You use the **sysctl** command directly. The **-p** option causes **sysctl** to read parameters from the **/etc/sysctl.conf** file (you can specify a different file). You can use the **-w** option to change specific parameters. You reference a parameter with its key. A *key* is the parameter name prefixed with its **proc** system categories (directories), such as **net.ipv4.ip_forward** for the **ip_forward** parameter located in **/proc/sys/net/ipv4/**. To display the value of a particular parameter, just use its key. The **-a** option lists all available changeable parameters. In the next example, the user changes the domain name parameter, referencing it with the **kernel.domainname** key (the **domainname** command also sets the **kernel.domainname** parameter):

```
# sysctl -w kernel.domainname="mytrek.com"
```

The following example turns on IP forwarding:

```
# sysctl -w net.ipv4.ip_forward=1
```

If you use just the key, you display the parameter's current value:

```
# sysctl net.ipv4.ip_forward
 net.ipv4.ip_forward = 1
```

Installing a New Kernel Version

To install a new kernel, you need to download the software packages for that kernel to your system. You can install a new kernel either by downloading a binary version from your distribution's website and installing it or by downloading the source code, compiling the kernel, and then installing the resulting binary file along with the modules. The binary version of the kernel is provided in an RPM package, and you can install a new kernel just as you would any other RPM software package.

The easiest way to install a new kernel is to use your distribution software install utility to automatically download and install a distribution prepared kernel package. The installation will create a GRUB entry so that when you boot, the new kernel will be listed as one of the options, usually the default.

If you want to download kernel RPM packages manually, keep in mind that the complete kernel installation usually includes a series of RPM packages, all beginning with the word *kernel*. There are also other packages you may need, which contain updated system configuration files used by the new kernel. You can use the packages already installed on your system as a guide. Use the **rpm** command with the **-qa** option to list all packages and then pipe that list through the **grep** command with the **kernel** pattern to display only the kernel packages:

```
rpm -qa | grep kernel
```

The kernel source code version is available for download from distribution FTP sites in the source directory and is included on distribution source code CD-ROM. You can also download the latest source directly from **kernel.org**. Wherever you download a kernel version from, it is always the same. The source code downloaded for a particular kernel version from a distribution site is the same as the one for **kernel.org**. Patches for that version can be applied to any distribution.

CPU Kernel Packages

Distributions will provide different kernel packages optimized for various popular CPUs. Choose the appropriate one for your machine. All the kernels include multiprocessor support. The x86 distribution will include the x86 versions, and the 64-bit distribution will hold the x86_64 versions. Each package is named kernel, but each has a different qualifier. For the x86, there are two different kernel packages, one for the newer Pentium 2, 3, and 4 CPUs, and one for the older Pentiums, AMD K6 CPUs, and other older systems. Each package will have a CPU reference in its filename: 686 for Pentium 2, 3, and 4; 586 for Pentium, K6, and other systems. For 64-bit systems, like the Athlon 64 series, the 64-bit distribution will include only an x86_64 package. There are also kernel packages for the Xen virtualization kernel, and there is the PAE kernel for extended system memory support. The kernel-kdump is a new minimal version for system dumps, useful for debugging (kdump support is included in 32-bit kernels).

For each kernel, there are usually corresponding kernel header packages (also known as builds), denoted with the term **devel**, that contain only the kernel headers. These are used for compiling kernel modules or software applications that do not need the full kernel source code, just the headers. The headers for your current kernel are already installed. The kernel headers will be installed in the **/etc/src/kernels** directory with a **build** link to it in the kernel's **/lib/modules** directory.

Installing Kernel Packages: /boot

On most distributions, kernels are installed in the **/boot** directory. Performing an `ls -l` operation on this directory lists all the currently installed kernels. A file for your old kernel and a file for your new one now exist. If you took the precautions described in the preceding section, you may have already renamed the older kernel. If you are using a boot loader such as GRUB, you need not change its configuration file (**/boot/grub/menu.lst**) to add the entry to invoke the new kernel. The kernel boots using the selected **/boot/vmlinuz-***version* kernel file. In your **/boot/grub/menu.lst** file, you need a kernel line to reference this kernel file. You also need to include a line for the RAM disk, `initrd`.

```
kernel /boot/vmlinuz-version ro root=/dev/hda3
initrd /boot/initrd-version.img
```

If your system has a SCSI controller or any other specialized hardware, RPM will also create a RAM disk to hold appropriate support modules (you can create a RAM disk manually with the `mkinitrd` command). The RAM disk is named **initrd-***kernel-version***.img** and is located in the **/boot** directory, as in **/boot/initrd-2.6.***version***.img**.

TIP *User-mode Linux (UML) is an optional version of the kernel designed to run as a stand-alone program separate from the kernel. In effect, it creates a virtual machine with disk storage implemented on a user file. UML is often used to test software or experiment with kernel configurations without harming the real system. You can also use UML to implement virtual hosting, by running several virtual machines on one physical host. With a virtual machine, you can control the access to the host system, providing greater security. You can find out more about user-mode Linux at* ***user-mode-linux.sourceforge.net****.*

Precautionary Steps for Modifying a Kernel of the Same Version

If you want to modify your kernel configuration and build a new one, you should retain a copy of your current kernel. In case something goes wrong with your modified version, you can always boot from the copy you kept. You do not have to worry about this happening if you are installing a new version of the kernel. New kernels are given different names, so the older one is not overwritten.

To retain a copy of your current kernel, you can make a backup copy of it, letting the original be overwritten. An installed version of a kernel makes use of several files in the **/boot** directory. Each file ends with that kernel version's number. These include the **vmlinuz** file, which is the actual kernel image file, along with several support files, **System.map**, **config**, and **initrd**. This **System.map** file contains kernel symbols needed by modules to start kernel functions. For example, the kernel image file is called **vmlinuz-***version,* where *version* is the version number attached, as in **vmlinuz-2.6.***version.* The **System.map** file for this kernel called **System.map-2.6.***version.* Here are the kernel files for version **2.6.v**:

```
/boot/vmlinuz-2.6.version
/boot/System.map-2.6.version
/boot/initrd-2.6.version.img
/boot/config-2.6.version
```

If, on the other hand, you are creating a modified version of the same kernel, the kernel file, here called **vmlinuz-2.6.***version,* will be overwritten with the new kernel image file, along with the **System.map** and **config** files. To keep your current working version, you first have to make a copy of these files. You make a copy of the **/boot/vmlinux-2.6.***version* file, giving it another name, as shown here:

```
cp /boot/vmlinuz-2.version /boot/vmlinuz-2.6.version.orig
```

You also make backups of the **System.map** and **config** files. You should also back up your modules located in the **/lib/modules/***version* directory, where *version* is the version number of the kernel. Otherwise, you will lose the modules already set up to work with the original kernel. For version 2.6.*version,* the libraries are located in **/lib/modules/2.6.***version.* If you are compiling a different version, those libraries are placed in a new directory named with the new version number.

Boot Loader

Installation of a kernel package will automatically create a GRUB boot loader entry for the new kernel. You will be able to select it on startup. Entries for your older kernel will remain.

Should you create a customized version of the current kernel while keeping the original versions as a backup, you would then need to create a new entry for the original kernel in the boot loader configuration file. Leaving the entry for the original kernel is advisable in case something goes wrong with the new kernel. This way, you can always reboot and select the original kernel. For example, in **/boot//grub/menu.lst**, add a new entry, similar to the one for the old kernel, which references the new kernel in its kernel statement. The **menu.lst** entry will look something like the following code. You could then select the entry with the title "Old Linux (2.6.*version*.orig)" at the GRUB menu to launch the original kernel.

```
title Original Linux (2.6.version.orig)
 root (hd0,2)
 kernel /boot/vmlinuz-2.6.version.orig root=/dev/hda3
 initrd /boot/initrd-2.6.version.orig.img
```

If you use a label for the boot partition, the **root** option for the **kernel** statement will look like this for a boot partition labeled /.

```
kernel /boot/vmlinuz-2.6.version.orig ro root=LABEL=/
```

Boot Disk

You should also have a boot CD-ROM ready, just in case something goes wrong with the installation (normally you'll have created one during installation). With a boot CD-ROM, you can start your system without using the boot loader. On Red Hat, Fedora, and similar distributions, you can create a boot CD-ROM using the **mkbootdisk** utility, but you need to know the full version number for your kernel. You can, in fact, have several kernels installed and create boot CD-ROMs for each one (your **/boot/grub/menu.lst** file lists your kernel version number). If the kernel version is 2.6.*version*, use it as the argument to the **mkbootdisk** command to create a boot CD-ROM for your system.

To make a boot CD-ROM, you can use the **--iso** option with the **--device** option to specify the CD image file. You can then burn the image file to a CD-ROM with an application like K3b. In the next example, the user creates a CD-ROM image file, called **myimage.iso**, for a boot CD-ROM of the 2.6.*version* kernel:

```
mkbootdisk --iso --device myimage.iso  2.6.version
```

Compiling the Kernel from Source Code

Instead of installing already-compiled binary versions of the kernel, you can install the kernel source code on your system and use it to create the kernel binary files yourself. Kernel source code files are compiled with the **gcc** compiler just as any other source code files are. One advantage to compiling the kernel is that you can enhance its configuration, adding support for certain kinds of devices such as Bluetooth devices.

NOTE *Many distributions can install kernel sources using their own software packaging methods. This is especially true for Fedora and Red Hat, whose SRPMS kernel packages install to the* ***/usr/src/redhat*** *directory and use SPEC files to extract the kernel version you want. Check your distribution's documentation on kernel source packages for details.*

Installing Kernel Sources: Kernel Archives and Patches

You can also download the original kernel source form **kernel.org**. This version will not be optimized for your distribution (distribution kernel packages will be optimized). It should be placed in the directory of your choosing, but not in the **/usr/src/linux** directory.

These versions are normally much more recent than those available on your distribution site, but they may not have been thoroughly tested on the distribution platform. The kernel source is in the form of compressed archives (**.tar.gz**). They have the prefix **linux** with the version name as the suffix. You decompress and extract the archive with the following commands. You first change to the local directory you chose and then unpack the archive with either File Roller or the **tar** command. It creates a directory with the prefix **linux** where the source files are placed. The following example extracts the 2.6.21.1 kernel:

```
cd mykernel
tar -xzvf linux-2.6.21.1.tar.gz
```

Be sure to unpack the archive for the **kernel.org** version in a directory you choose, such as **mykernel** in a home directory. The source will reside within a subdirectory that has the prefix **linux** and a suffix consisting of the kernel version, as in **linux-2.6.21** for kernel 2.6, revision 21. The local directory in this example would be **mykernel/linux-2.6.21**.

TIP *If you are using the original kernel source, you should also check for any patches.*

Configuring the Kernel

Once the source is installed, you must configure the kernel. Configuration consists of determining the features for which you want to provide kernel-level support. These include drivers for different devices, such as sound cards and SCSI devices. You can configure features as directly included in the kernel itself or as modules the kernel can load as needed. You can also specifically exclude features. Features incorporated directly into the kernel make for a larger kernel program. Features set up as separate modules can also be easily updated. Documentation for many devices that provide sound, video, or network support can be found in the **/usr/share/doc** directory. Check the kernel-doc package to find a listing of the documentation provided.

NOTE *If you configured your kernel previously and now want to start over from the default settings, you can use the* ***make mrproper*** *command to restore the default kernel configuration.*

Kernel Configuration Tools

You can configure the kernel using one of several available configuration tools: **config**, **menuconfig**, **xconfig** (qconf), and **gconfig** (gkc). You can also edit the configuration file directly. These tools perform the same configuration tasks but use different interfaces. The **config** tool is a simple configure script providing line-based prompts for different configuration options. The **menuconfig** tool provides a cursor-based menu, which you can still run from the command line. Menu entries exist for different configuration categories, and you can pick and choose the ones you want. To mark a feature for inclusion in the kernel, move to it and press the SPACEBAR. An asterisk appears in the empty parentheses to the left of

the entry. If you want to make it a module, press M and an *M* appears in the parentheses. The **xconfig** option runs qconf, the QT (KDE)–based GUI kernel configuration tool, and requires that the QT libraries (KDE) be installed first. The **gconfig** option runs the gkc tool, which uses a GTK interface, requiring that GNOME be installed first. Both qconf and gkc provide expandable menu trees, selectable panels, and help windows. Selectable features include check buttons you can click. All these tools save their settings to the **.config** file in the kernel source's directory. If you want to remove a configuration entirely, you can use the **mrproper** option to remove the **.config** file and any binary files, starting over from scratch.

```
make mrproper
```

You start a configuration tool by preceding it with the **make** command. Be sure you are in the kernel directory (either the distribution's location for its kernel packages, or the local directory you used for the compressed archive, such as **tar.gz**). The process of starting a configuration tool is a **make** operation that uses the Linux kernel Makefile. The **xconfig** tool should be started from a terminal window on your window manager. The **menuconfig** and **config** tools are started on a shell command line. The following example lists commands to start **xconfig**, **gconfig**, **menuconfig**, and **config**:

```
make gconfig
make xconfig
make menuconfig
make config
```

gconfig (gkc)

The GTK kernel configuration tool (gkc) is invoked with the **gconfig** option. This uses a GNOME-based interface that is similar to qconf (**xconfig**). The gkc tool opens a Linux Kernel Configuration window with expandable submenus like those for qconf. Many categories are organized into a few major headings, with many now included under the Device Drivers menu. The Load and Save buttons and File menu entries can be used to save the configuration or to copy it to a file. Single, Split, and Full view buttons let you display menus in one window, in a display panel with another panel to containing an expandable tree to select entries, or as a single expandable tree of entries. The Expand button will expand all headings and subheadings, whereas Collapse will let you expand only those you want displayed. Use the down and side triangles for each entry to expand or collapse subentries.

Clicking an entry opens a window that lists different features you can include. Entries are arranged in columns listing the option, its actual name, its range (yes, module, or no), and its data (yes, no, or module status). Entries in the Options menu let you determine what columns to display: Name for the actual module name; Range for the selectable yes, no, and module entries; and Data for the option status, titled as Value.

The Range entries are titled N, M, and Y and are used to select whether not to include an option (N), to load it as a module (M), or to compile it directly into the kernel (Y). Entries that you can select will display an underscore. Clicking the underscore will change its entry to Y for module or direct kernel inclusion, and N for no inclusion. The Value column will show which is currently selected.

The Options column will include a status showing whether the option is included directly (check mark), included as a module (line mark), or not included at all (empty).

To quickly select or deselect an entry, double-click the option name in the Options field. You will see its check box checked, lined (module), or empty. Corresponding N, M, and Y entries for no inclusion, module, or kernel inclusion are selected. The default preference for either module or direct kernel inclusion for that option is selected automatically. You can change it manually if you wish.

xconfig (qconf)

The **`xconfig`** option invokes the qconf tool, which is based on KDE QT libraries. KDE has to first be installed. The qconf tool opens a Linux Kernel Configuration window listing the different configuration categories. It has a slightly simpler interface, without the expand or collapse buttons or the columns for module and source status.

Important Kernel Configuration Features

The **`xconfig`**, **`menuconfig`**, and **`gconfig`** tools provide excellent context-sensitive help for each entry. To the right of each entry is a Help button. Click it to display a detailed explanation of what that feature does and why you would include it either directly or as a module, or even exclude it. When you are in doubt about a feature, always use the Help button to learn exactly what it does and why you would want to use it. Many of the key features are described here. The primary category for a feature is listed in parentheses.

TIP *As a rule, features in continual use, such as network and file system support, should be compiled directly into the kernel. Features that could easily change, such as sound cards, or features used less frequently should be compiled as modules. Otherwise, your kernel image file may become too large and will be slower to run.*

- **Loadable Module Support** In most cases, you should make sure your kernel can load modules. Click the Loadable Module Support button to display a listing of several module management options. Make sure Enable Loadable Module Support is marked Yes. This feature allows your kernel to load modules as they are needed. Kernel Module Loader should also be set to Yes, because this allows your daemons, such as your web server, to load any modules they may need.
- **Processor Type And Features** The Processor Type And Features window enables you to set up support for your particular system. Here, you select the type of processor you have (486, 586, 686, Pentium III, Pentium IV, and so forth), as well as the amount of maximum memory your system supports (up to 64GB with the 2.4 kernel).
- **General Setup** The General Setup window enables you to select general features, such as networking, PCI BIOS support, and power management, as well as support for ELF and **a.out** binaries. Also supported is **`sysctl`** for dynamically changing kernel parameters specified in the **/proc** files. You can use **redhat-config-proc** (the Kernel Tuning tool in the System Tools menu) to make these dynamic changes to the kernel. In the additional device driver support menu, you can enable specialized features like Crypto IP Encapsulation (CIPE) and accelerated SSL.

- **Block Devices (Device Drivers)** The Block Devices window lists entries that enable support for your IDE, floppy drive, and parallel port devices. Special features, such as RAM disk support and the loopback device for mounting CD-ROM image files, are also there.
- **Multi-Device Support (RAID and LVM) (Device Drivers)** The Multi-Device Support window lists entries that enable the use of RAID devices. You can choose the level of RAID support you want. Here you can also enable Logical Volume Management support (LVM), which lets you combine partitions into logical volumes that can be managed dynamically.
- **Networking Options (Device Drivers/Networking Support)** The Networking Options window lists an extensive set of networking capabilities. The TCP/IP Networking entry must be set to enable any kind of Internet networking. Here, you can specify features that enable your system to operate as a gateway, firewall, or router. Network Packet Filtering enables support for an IPtables firewall. Support also exists for other kinds of networks, including AppleTalk and IPX. AppleTalk must be enabled if you want to use NetTalk to connect to a Macintosh system on your network (Filesystems).
- **ATA/IDE/MFM/RLL Support (Device Drivers)** In the ATA/IDE/MFM/RLL Support window, you can click the IDE, ATA, and ATAPI Block Device button to open a window where you can select support for IDE ATA hard drives and ATAPI CD-ROMs.
- **SCSI Support (Device Drivers)** If you have any SCSI devices on your system, make sure the entries in the SCSI Support window are set to Yes. You enable support for SCSI disks, tape drives, and CD-ROMs here. The SCSI Low-Level Drivers window displays an extensive list of SCSI devices currently supported by Linux. Be sure the ones you have are selected.
- **Network Device Support (Device Drivers/Networking Support)** The Network Device Support window lists several general features for network device support. There are entries here for windows that list support for particular types of network devices, including Ethernet devices, token ring devices, WAN interfaces, and AppleTalk devices. Many of these devices are created as modules you can load as needed. You can elect to rebuild your kernel with support for any of these devices built directly into the kernel.
- **Multimedia Devices (Device Drivers)** Multimedia devices provide support for various multimedia cards as well as Video4Linux.
- **File Systems** The File Systems window lists the different types of file systems Linux can support. These include Windows file systems such as DOS, VFAT, and NTFS, as well as CD-ROM file systems such as ISO and UDF. Network file systems such as NFS, SMB (Samba), and NCP (NetWare) are included, as well as miscellaneous file systems such as HFS (Macintosh).
- **Character Devices (Device Drivers)** The Character Devices window lists features for devices such as your keyboard, mouse, and serial ports. Support exists for both serial and bus mice.

- **Sound (Device Drivers)** For the 2.4 kernel, the Sound window lists different sound cards supported by the kernel. Select the one on your system. For older systems, you may have to provide the IRQ, DMA, and Base I/O your sound card uses. These are compiled as separate modules, some of which you can elect to include directly in the kernel if you want. For the 2.6 kernel, you can select the Advanced Linux Sound Architecture sound support, expanding it to the drivers for particular sound devices (the Open Sound System is also included, though deprecated).
- **Bluetooth Devices (Device Drivers/Networking Support)** Support is here for Bluetooth-enabled peripherals, listing drivers for USB, serial, and PC card interfaces.
- **Kernel Hacking** The Kernel Hacking window lists features of interest to developers who work at the kernel level and need to modify the kernel code. You can have the kernel include debugging information and also provide some measure of control during crashes.

Once you set your options, save your configuration. Selecting the Save entry on the File menu overwrites your **.config** configuration file. The Save As option lets you save your configuration to a particular file.

Compiling and Installing the Kernel

Now that the configuration is ready, you can compile your kernel. You will have to clean up any object and dependency files that may remain from a previous compilation. Use the following command to remove such files:

```
make clean
```

You can use several options to compile the kernel (see Table 32-1). The **`bzImage`** option simply generates a kernel file called **bzImage** and places it in the **arch** directory. For Intel

Option	Description
`zImage`	Creates the kernel file called **zImage** located in the **arch** or **arch/i386/boot** directory.
`install`	Creates the kernel and installs it on your system.
`zdisk`	Creates a kernel file and installs it on a floppy disk (creates a boot disk, 1.44 MB).
`bzImage`	Creates the compressed kernel file and calls it **bzImage**.
`bzdisk`	Creates the kernel and installs it on a floppy disk (creates a boot disk). Useful only for smaller kernel builds, 1.44 MB.
`fdimage`	Creates floppy disk image with the kernel, 1.44 MB (bootable).
`fdimage288`	Creates floppy disk 2.88 image with the kernel (bootable).

TABLE 32-1 Compiling Options for Kernel **`make`** Command

and AMD systems, you find **bzImage** in the **i386/boot** subdirectory, **arch/i386/boot**. For a kernel source, this is in **arch/i386/boot**.

```
make bzImage
```

The options in Table 32-1 create the kernel, but not the modules—those features of the kernel to be compiled into separate modules. To compile your modules, use the **make** command with the **modules** argument.

```
make modules
```

The **make** command without arguments will create the **bzImage** and the modules.

```
make
```

To install your modules, use the **make** command with the **modules_install** option. This installs the modules in the **/lib/modules/***version-num* directory, where *version-num* is the version number of the kernel. You should make a backup copy of the old modules before you install the new ones.

```
make modules_install
```

The **install** option both generates the kernel files and installs them on your system as **vmlinuz**, incorporating the **make bzImage** step. This operation will place the kernel files such as **bzImage** in the **/boot** directory, giving them the appropriate names and kernel version numbers.

```
make install
```

The commands for a simple compilation and installation are shown here:

```
make clean
make
make modules_install
make install
```

If you want, you can enter all these on fewer lines, separating the commands with semicolons, as shown here:

```
make clean; make; make modules_install; make install
```

A safer way to perform these operations on single lines is to make them conditionally dependent on one another, using the **&&** command. In the preceding method, if one operation has an error, the next one will still be executed. By making the operations conditional, each operation is run only if the preceding one is successful.

```
make clean && make && make modules_install &&  make install
```

Installing the Kernel Image Manually

To install a kernel **bzImage** file manually, copy the **bzImage** file to the directory where the kernel resides and give it the name used on your distribution, such as **vmlinuz-2.6.***version*. Remember to first back up the old kernel file, as noted in the precautionary steps. **vmlinuz** is a symbolic link to an actual kernel file that has the term **vmlinuz** with the version name.

So, to manually install a **bzImage** file, you copy it to the **/boot** directory with the attached version number such as **vmlinuz-2.6.***version.*

```
make bzImage
cp arch/i386/boot/bzImage /boot/vmlinuz-2.6.version
```

TIP *The* **bzImage** *option, and those options that begin with the letter b, create a compressed kernel image. This kernel image may not work on older systems. If not, try using the* **zImage** *option to create a kernel file called* ***zImage****. Then install the* ***zImage*** *file manually the same way as you would do with* ***bzImage****. Bear in mind that support for* **zImage** *will be phased out eventually.*

You will also have to make a copy of the **System.map** file, linking it to the **System.map** symbolic link.

```
cp arch/i386/boot/System.map  /boot/System.map-2.6.version
```

The following commands show a basic compilation and a manual installation. First, all previous binary files are removed with the **clean** option. Then the kernel is created using the **bzImage** option. This creates a kernel program called **bzImage** located in the **arch/i386/boot** directory. This kernel file is copied to the **/boot** directory and given the name **vmlinuz-2.6.***version.* Then a symbolic link called **/boot/vmlinuz** is created to the kernel **vmlinuz-2.6.***version* file. Finally, the modules are created and installed:

```
make clean
make
make modules_install
cp arch/i386/boot/bzImage /boot/vmlinuz-2.6.version
cp System.map /boot/System.map-2.6.version
```

Kernel Boot Disks

Instead of installing the kernel on your system, you can simply place it on a boot disk or CD-ROM and boot your system from that disc. For a CD-ROM you can first create the kernel as a **bzImage**, install the kernel, and then use **mkbootdisk** to create a bootable CD-ROM. For a boot disk you have the option of creating either a floppy disk directly or a floppy disk image.

If you are using a stripped-down configured version of the kernel that will fit on a 1.44 MB floppy disk, you can use the **bzdisk** or **zdisk** options to compile the kernel and install directly on a floppy. You will need a floppy disk placed in your floppy drive. A standard kernel 2.6 configuration is too large to fit on a floppy, though 2.4 versions will.

For a floppy disk image you can create either a 1.44 or 2.88 image (which will hold the 2.6 kernel). Use the **fdimage** option for a 1.44 image and **fdimage288** for the 2.88 image. Both **fdimage** and **fimage288** create corresponding floppy disk images in the **arch/i386/boot** directory. They use their own **mtools.conf** configuration located in that directory to generate the letters for the floppy disk image, which **mcopy** can then use to create the images. The **fdimage288** image is often used for virtual users.

```
make bzdisk
make fdimage
make fdimage288
```

TIP *If you are experimenting with your kernel configurations, it may be safer to put a new kernel version on a bootable CD-ROM, rather than installing it on your system. If something goes wrong, you can always boot up normally with your original kernel still on your system (though you can always configure your boot loader to access previous versions).*

Boot Loader Configurations: GRUB

If you are using a boot loader such as GRUB or LILO, you can configure your system to enable you to start any of your installed kernels. As seen in the earlier section "Precautionary Steps for Modifying a Kernel of the Same Version," you can create an added entry in the boot loader configuration file for your old kernel. As you install new kernel versions, you can simply add more entries, enabling you to use any of the previous kernels. Whenever you boot, your boot loader will then present you with a list of kernels to choose from. For example, you can install a developmental version of the kernel, along with a current stable version, while keeping your old version. In the image line for each entry, you specify the filename of the kernel. You can create another boot loader entry for your older kernel.

In the next example, the Grub configuration file (**/boot/grub/menu.lst**) contains entries for two Linux kernels, one for the kernel installed earlier, **2.6.20.3**, and one for a more recent kernel, **2.6.21.1**. With GRUB, you only have to add an new entry for the new kernel.

```
#
#boot=/dev/hda
default=0
timeout=30
splashimage=(hd0,2)/boot/grub/splash.xpm.gz
title New Linux (2.6.21.1)
   root (hd0,2)
   kernel /boot/vmlinuz-2.6.21.1 ro root=/dev/hda3
   initrd /boot/initrd-2.6.21.1.img
title  Old Linux (2.6.20.3)
   root (hd0,2)
   kernel /boot/vmlinuz-2.6.20.3 ro root=/dev/hda3
   initrd /boot/initrd-2.6.20.3.img
title Windows XP
   rootnoverify (hd0,0)
   chainloader +1
```

Should your root directory be installed on a logical LVM partition, the LVM name is used such as Logical01 volume on the Logical00 volume group. Your **kernel** entry will look something like this.

```
kernel /vmlinuz-2.6.21.1 ro root=/dev/Logical00/Logical01
```

Module RAM Disks

If your system uses certain block devices unsupported by the kernel, such as some SCSI, RAID, or IDE devices, you will need to load certain required modules when you boot. Such block device modules are kept on a RAM disk that is accessed when your system first starts up (RAM disks are also used for diskless systems). For example, if you have a SCSI hard

drive or CD-ROMs, the SCSI drivers for them are often held in modules that are loaded whenever you start up your system. These modules are stored in a RAM disk from which the startup process reads. If you create a new kernel that needs to load modules to start up, you must create a new RAM disk for those modules. You need to create a new RAM disk only if your kernel has to load modules at startup. If, for example, you use a SCSI hard drive but you incorporated a SCSI hard drive and CD-ROM support (including support for the specific model) directly into your kernel, you don't need to set up a RAM disk (support for most IDE hard drives and CD-ROMs is already incorporated directly into the kernel).

If you need to create a RAM disk, you can use the **mkinitrd** command to create a RAM disk image file. The **mkinitrd** command incorporates all the IDE, SCSI, and RAID modules that your system uses, including those listed in your **/etc/modules.conf** file. See the Man pages for **mkinitrd** and RAM disk documentation for more details. **mkinitrd** takes as its arguments the name of the RAM disk image file and the kernel that the modules are taken from. In the following example, a RAM disk image called **initrd-2.**6.*version***.img** is created in the **/boot** directory, using modules from the 2.6.21 kernel. The 2.6.21 kernel must already be installed on your system and its modules created.

```
# mkinitrd /boot/initrd-2.6.version.img 2.6.21.1
```

You can select certain modules to be loaded before or after any SCSI module. The **--preload** option loads before the SCSI modules, and **--with** loads after. For example, to load RAID5 support before the SCSI modules, use **--preload=raid5**:

```
mkinitrd --preload=raid5 raid-ramdisk 2.6.21.1
```

In the **/etc/grub/menu.lst** segment for the new kernel, place an **initrd** entry specifying the new RAM disk:

```
initrd /boot/initrd-2.6.21.1.img
```

Virtualization

There are now several methods of virtualization available for use on Linux. These range from the para-virtualization implementation employed by Xen to the hardware acceleration used by the Kernel-based Virtualization Machine (KVM) for Intel and AMD processors with hardware virtualization support. You can even use software emulation. All of these can be installed and managed easily with the Virtual Machine Manager, a GNOME-based tool that provides a simple GUI interface for managing your virtual machines and installing new ones. Linux also provides the GNOME VM applet, **gnome-applet-vm**, a panel applet that can monitor your virtual machines. See Table 32-2 for a listing of virtualization resources. See **virt.kernelnewbies.org** for general virtualization links and overview.

All virtualization methods can be installed and managed with the Virtual Machine Manager. The Virtual Machine Manager greatly simplifies the process of installing and managing virtual operating systems (guest OSs). With just a few steps you can install Windows or other Linux distributions on your Linux system and run them as guest operating systems whenever you need them. KVM virtual hosts will run directly from the processor, running almost as fast and as stable as if you installed it separately with a dual boot configuration.

Resource	Description
virt.kernelnewbies.org	Virtualization kernel documentation
virt-manager.et.redhat.com/	Virtual Machine Manager, virt-manager
xensource.com	Xen para-virtualization website
fabrice.bellard.free.fr/qemu/	QEMU software virtualization
kvm.qumranet.com/kvmwiki	KVM hardware virtualization
libvirt.org/	vibvirt tool kit for accessing Linux virtualization capabilities

Table 32-2 Virtualization Resources

There are two major methods currently used for virtualization, full and para-virtualization. Full virtualization (KVM or QEMU) runs a guest OS independently, whereas para-virtualization (Xen) requires that you first boot up a Xen Linux kernel from which to launch para-virtualized guest OS systems. This means that a fully virtualized OS can be started with the Virtual Machine Manager from a normal Linux kernel, whereas a para-virtualized OS requires booting up with a Xen kernel.

Virtual Machine Manager: virt-manager (Red Hat)

The Virtual Machine Manager is a project developed and supported by Red Hat and currently available on many similar distributions. You can easily manage and set up virtual machines using the Virtual Machine Manger (**virt-manager**). Be sure that **virt-manger** is installed. This will display a window listing your virtual machines. Features like the machine ID, name, status, CPU, and memory usage will be displayed. You can use the View menu to determine what features to display. Click the Help entry in the Help menu to show a detailed manual for Virtual Machine Manager.

For detailed information about the host machine, click Host Details from the Edit menu. The Overview panel will show information like the hostname, the number of CPUs it has, and the kind of hypervisor it can launch. The Virtual Network panel shows your virtual networks, listing IPv4 connection information, the device name, and the network name. A default virtual network will already be set up. Select guest and click Details button for guest information.

To create a virtual machine, select New Machine from the File menu. This will start up **virt-install** wizard. You will be prompted for the name, the kind of virtualization, the location of the operating system install disk or files, the storage to use for the guest operating system, and the amount of system memory to allocate for the guest OS.

After entering a name, you choose your virtualization method. If you are running a standard kernel, you will have only the option to use a fully virtualized method. On systems with Intel VT and AMD SVM processors you will also have the option to enable hardware acceleration. This means using KVM (Kernel-based Virtualization Machine) support that will provide processor-level hardware virtualization. For processors without hardware virtualization support, a software emulation is used.

If, instead, you are running Virtual Machine Manager from the Xen kernel (as Domain 0), you can use para-virtualization. For versions of a guest OS specially modified for use by

Xen, that guest OS can be run with virtualization employed as required. In addition, for Intel VT and AMD SVM, Xen HVM methods can be used to employ hardware virtualization when virtualization is needed.

TIP *With a system with extensive memory and processor support, you can even run or install guest operating systems simultaneously using KVM from the Virtual Machine Manager.*

You then choose the location of the OS install media. For a fully virtualized OS, this can either be a disk image or a CD/DVD-ROM, such as a Windows install disk. You then choose the type of operating system you are installing, first selecting a category like Linux or Windows, and then a particular distribution or version like Centos Linux or Windows XP. For a para-virtualized OS (Xen), you choose a network location for the install media.

You then choose the storage method. This can be either an existing partition or a file. If you choose a file, you can either set a fixed size (like a fixed partition), or have the file expand as needed. Should the file be on a partition with a great deal of free space, this may not be an issue. Initially the file will be 4GB, though you may want to make it larger to allow for regular use.

You then choose a virtual network or a physical device for your network connection. Then choose the amount of system memory to allocate to each virtual machine, as well as the number of virtual CPUs to use. A final screen displays all your configuration information for the new virtual machine before you start installation. You can still cancel at this point. When you start installation, the install window for the guest OS is displayed and you install as you normally would. An installed OS is run in the virtual machine console window. There are buttons to run, pause, and shut down the OS.

TIP *You can also manage your virtual machines from the command line with* **`virsh`**.

Kernel-Based Virtualization Machine (KVM): Hardware Virtualization

With kernel version 2.6.21, hardware virtualization is now directly supported in the kernel (pervious versions used a kernel module). Hardware virtualization is implemented by Intel and AMD as a Hardware Virtual Machine abstraction layer. Intel processors that have hardware virtualization support are labeled VT (Virtualization Technology), and AMD processors are labeled SVM (Secure Virtual Machine). An HVM system has the capability to provide full virtualization, not requiring a specially modified versions of an OS kernel like Xen's para-virtualization method uses. You can even run Windows XP directly from Linux using the HVM capability. KVM is a open source project developed by Qumranet, **kvm .qumranet.com/kvmwiki**. Be sure to also check for KVM information in the Virtualization documentation for your distribution.

TIP *KVM is run with a modified version of QEMU, which has limited virtual device support, such as the graphics driver (Xen has full native device driver access).*

KVM uses the hardware virtualization in a processor to run a virtual machine directly from hardware. There is no underlying software translation; whereas Xen will work through an underlying domain 0 kernel, KVM operates directly with the processor.

Hardware requirements are as follows:

- An Intel (VT) or AMD (SVM) virtualization enabled processor (such as AMD AM2 socket processors or Intel Core2Duo processors). You may need to enable virtualization support in your motherboard. Some motherboards will work better than others. In some cases, you may have to disable ACPI support in the motherboard BIOS to allow Windows XP to run.
- At least 1 GB of system memory to allow space for the virtual OS to run. The hardware virtual OS requires its own memory.

KVM is launched as a process directly from the Linux kernel, as if booting to a new OS. It can be managed like any Linux process. KVM adds a quest process mode with its own user and kernel mode. This is in addition to the Linux kernel and user modes. KVM uses its own device driver to interface with the processor's virtualization hardware, **/dev/kvm**. KVM uses the kernel modules kvm-intel or kvm-amd to interface with the processor's virtualization hardware. A modified version of a software emulator QEMU is used to run the OS guest. QEMU was originally designed as an emulator and is also available as such for processors without hardware virtualization. See **fabrice.bellard.free.fr/qemu** for more information on QEMU. You do not have to install Xen to run KVM; KVM can be run separately, though installing Xen will not interfere with KVM installs.

NOTE *You can implement KVM manually using QEMU, a processor emulator for full virtualization. First you create an image file for the new OS with* **`qemu-img`**. *Then use* **`kvm`** *to start the guest OS.*

Be sure to boot into the standard kernel, not into the Xen kernel. Start the Virtual Machine Manager on your Gnome desktop. Choose New Machine for the File menu. This starts the **virt-install** wizard. When choosing the type of virtualization to use, select Fully Virtualized and make sure hardware acceleration is selected (Enabled kernel/hardware acceleration). You are then prompted for various features like the name, the amount of system memory to use, whether to use a given partition or an image file along with the file size, graphics support, and where the install image is located (this can be a CD/DVD-ROM, though a disk image is preferred for Windows).

Once installed, you can use the Virtual Machine Manager to start up your guest OS at any time. Your guest OS is run in a virtual machine console.

NOTE *To access data directly on your virtual disks or files, you can use* **`lomount`** *or* **`kpartx`**.

Xen Virtualization Kernel

Distributions will normally provide versions of the kernel that incorporate Xen Virtualization. Xen Virtualization technology allows you to run different operating systems on a Linux system, as well as run virtual versions of the kernel to test new applications. Xen is an open source project originally developed by the University of Cambridge Computer Laboratory in coordination with the Open Source Development Labs and several Linux distributors. You can find more about Xen at **cl.cam.ac.uk/Research/SRG/netos/xen**. Xen development is currently managed by Xen.source, a commercial service that provides both free open source

versions of Xen and commercial implementations with support. Here you can find detailed documentation on the latest Xen releases, **xensource.com**.

NOTE *VMware provides a free version of its virtualization server. You can also purchase the virtualization desktop to install other OSs, as well as the ESX virtualization server that is more stable and efficient.*

On a single Xen server you can run several virtual machines to run different operating systems at the same time. Commercial virtualization is currently provided by VMware, though its also provides a free version of its virtualization server. Xen is a para-virtualized system, meaning that the guest operation system has to be modified to run on Xen. It cannot run without modification as it can on a fully virtualized system like VMware. Xen uses a para-virtualization approach to increase efficiency, giving its virtual machines nearly the same level of efficiency as the native kernel. This makes virtualization practical for enterprise-level systems. Some of the advantages cited for Xen are setting up a separate test system, isolating servers in virtual machines on the same system, and letting virtual machine access the hardware support provided by the native kernel. For an operating system to work on Xen, it must be configured to access the Xen interface. Currently only Unix and Linux operating systems are configured to be Xen compatible, though work is progressing on Windows.

To use the Xen kernel, you first have to install the Xen kernel package as well as the Xen server, tools, and documentation. There is one Xen kernel package, which incorporates support for running Xen in domain 0 (xen0), as a server, and for unprivileged (xenU) user access. Detailed documentation will be in **/usr/share/doc/xen-***version*. Configuration files will be placed in **/etc/xen** directory, and corresponding kernels in the **/boot** and **/lib/modules** directories. In the **/etc/xen** directory you will find the **xend-config** file for configuring the Xen **xend** server, as well as example Xen configuration files.

Once the package is installed, reboot and select the Xen kernel from the GRUB screen. Your standard original kernel will also be listed, which you can select to return to a normal kernel. Selecting Xen will start up the Xen kernel with the ability to create your Xen-based virtual systems. Everything will look exactly the same. If you have the GNOME VM monitor applet running, it will now detect a 0 domain.

Xen sets up separate virtual machines called domains. When the Xen kernel starts up, it creates a primary domain, domain0, which manages your system and sets up virtual machines for other operating systems. Management of the virtual machines is handled by the **xend** server. Your native kernel is installed on domain0, which will handle most of the hardware devices for all the other virtual machines.

You control the domains with the **xend** server. **xend** messages are placed in the **/var/log/xend.log** file. The **xend** server should automatically be started when you start up with the Xen kernel.

NOTE *Xen also provides support for the Hardware Virtual Machine (HVM), the HVM abstraction layer that Intel is implementing in its new processors as Intel VT-x. AMD will implement HVM as SVM. The example configuration file for HVM in the* ***/etc/xen*** *directory has the extension* ***.hvm****. In this file, options are set to detect and use HVM. The* ***virt-install*** *script also checks for HVM.*

XenMan

On most distributions, including Debian, Fedora, SUSE, and Ubuntu, you can use XenMan to manage your Xen virtual machines (**xenman.sourceforge.net**). The XenMan tool provides a desktop interface from which you can manage Xen domains, adding new ones or deleting old ones. Dashboards let you check statistics like CPU and memory useage. You can manage each virtual machine, starting, stopping, or rebooting. You can even save a snapshot of a machine and restore to that point. To start XenMan, enter the **xenman** command from a terminal window. XenMan configurations file is **xenman.conf.**

Users have their own in the **.xenman** directory. Global definitions are held in the **/etc/xenman/xenman.conf** file. The configuration file defines paths for your virtual block devices, location of snapshots, and where the Xen configuration files are located. It also provides environment (current doman), any specific application configurations, and client configuration like GNOME support. XenMan also supports the use of images, collected into an image store. An image is a pre-defined virtual machine from which numerous other virtual machines could be generated. You could set up Fedora and Ubuntu images, and then generate several Fedora or Ubuntu virtual machines using those images.

virt-install (Red Hat)

On Red Hat, Fedora, and similar distributions, instead of configuring a file directly or using the Virtual Machine Manager, you can use the **virt-install** script. It currently can only install from a remote network location using an **http://**, **nfs://**, or **ftp://** prefix. This script will not allow you to use less than 256 MB for each virtual machine. If you want to use less memory than that, say for a scaled-down version of Linux, you will have to use the configuration files directly as described in the preceding section.

If you have a limited amount of RAM memory, you may need to limit the amount the **domain 0** virtual machine is using. You can reduce this to the recommended 256 MB with the following command:

```
xm mem-set 0 256
```

To start **virt-install** script, open a terminal window and enter the script name.

```
virt-install
```

You will be prompted to set different parameters, and then a configuration file will be automatically generated. You will first be prompted to name the virtual machine. This is your hostname. Then you are asked how much RAM to allocate. A minimum of 256 MB is required. This means that for just one virtual machine you have to have at least 500 MB of RAM, 256 MB for the Xen0 server (Domain 0) and 256 MB for the guest/user machine (Domain 1). More virtual machines would use correspondingly more RAM.

You are then prompted for the disk path for the virtual machine image file. Enter the path with the image filename. You are then prompted for the size of the image file in gigabytes. Virtual machines use an image file where its entire system is kept. Finally, you are prompted for the location of installation files for the operating system you want to install. Here you can enter an FTP, web, or NFS site. Keep in mind that the script looks for the Xen-compatible kernel images in the **images/Xen** directory of the distribution. Your download location has to prefix this directory. The online Fedora 7 directory is shown in the next example:

```
# virt-install
What is the name of your virtual machine? my-newvm1
How much RAM should be allocated (in megabytes)? 256
What would you like to use as the disk (path)? /home/my-newvm1
How large would you like the disk to be (in gigabytes)? 8
Would you like to enable graphics support?(yes or no) yes
What is the install location?
http://download.fedora.redhat.com/pub/fedora/linux/7/i386/os/
```

Once the files have been downloaded, the text-based install interface will start up, asking for keyboard and language. If you enabled graphics support, the standard graphical install will start up.

Creating Xen Virtual Machines with Configuration Files

Advanced users can create a virtual machine by setting up and editing the machine Xen configuration file directly. Instead of using the Virtual Machine Manager or **virt-install** to generate your configuration and install your OS, you can set more refined options. You first have to create the machine's configuration file in the **/etc/xen** directory. In this directory you will find sample configuration files you can use as a template. There are standard Xen configuration examples, as well as configuration for Xen HVM implementations. Here are settings you may want to change:

- **kernel** The path to the kernel image used by the virtual machine
- **root** The root device for the domain
- **memory** The amount of memory you will allow the domain to use
- **disk** The block devices (partitions) you want the domain to use
- **dhcp** Have the domain use DHCP to set networking; for manual configuration, you can set netmask and gateway parameters
- **hostname** The hostname for the virtual machine
- **vif** The MAC address to use (random ones will be generated if none is specified)
- **extra** Additional boot parameters
- **restart** Automatic restart options: always, never, onreboot

You will have to set the kernel. This is the location of the xenU kernel. The root entry specifies the partition where the boot image is stored. The disk entry lets you specify disks that the new domain will use. These can be logical volumes or disk image files. These cannot be already mounted by the primary domain. Note that **xenguest-install** described later will set up a disk image file to be used by the virtual machine, instead of a logical volume. The following uses a disk image **/home/mynewvm1**, which will appear to the virtual machine as **xvda** and have read/write access:

```
disk = [ 'phy:/home/mynewvm1,xvda,w' ]
```

You may also want to enable **dhcp** if your network uses that to set up a network connection. For example, the kernel setting would look like this:

```
kernel = "/boot/vmlinuz-2.6.21.1
```

You use the **xm** command with the **create** option followed by the configuration file. In this example there is a configuration file in **/etc/xen** called **my-unewvm1**, which is the same name as the virtual machine. Check the Man page for **xmdomain.cfg** for detailed configuration options and examples.

```
xm create my-newvm1
```

You can then connect to it with the **console** option.

```
xm console my-newvm1
```

You can combine the two **xm commands** using the **-c** option for the console connection.

```
xm create -c my-newvm1
```

You will still have to install the operating system you want to use. You can do this on a designated partition or logical volume, or use **virt-install**. You can install with a text-based system or use VNC for a virtual install.

Managing Xen Virtual Machines with xm

After you have installed the system, you can create a connection to it with the **xm** command. The **xm** command is available on all distributions. To access a particular domain, use the **console** option and the domain name.

```
xm console my-newvm1
```

If the domain no longer exists, you have to also create it and then connect.

```
xm create -c my-newvm1
```

Check the **xm** options on the **xm** Man page for other operations you can perform on your virtual machines.

To access domains, you use the **xm** command. The **list** option lists your domains. The listing will include detailed information such as its domain ID, the CPU time used, memory used, and the domain state. The following lists your domains:

```
xm list
```

The **xm save** and **restore** options can be used to suspend and restart a domain.

Block devices such as partitions and CD-ROMs can be exported from the main domain to virtual domains. This allows a given virtual domain to use a particular partition. You can even share block devices between domains, though such shared devices should be read only.

To start and stop your domains, you can use the **xendomains** service script. The **xendomains** script will use **xm** with the **create** option to create a domain configured in the **/etc/xen/auto** directory. Place Xen domain configuration files in this directory for **xendomains** to start. These domains will be started automatically when the kernel for domain0 is started. The following command manually starts your domains:

```
service xendomains start
```

NOTE *To more efficiently use block memory (hard disk partitions), you can implement dynamically allocated space using either file VBDs (virtual block devices implemented as files) or LVM VBDs.*

CHAPTER 33

Backup Management

Backup operations have become an important part of administrative duties. Several backup tools are provided on Linux systems, including Anaconda and the traditional dump/restore tools, as well as the **rsync** command for making individual copies. Anaconda provides server-based backups, letting different systems on a network back up to a central server. BackupPC provides network and local backup using configured rsync and tar tools. The dump tools let you refine your backup process, detecting data changed since the last backup. Table 33-1 lists websites for Linux backup tools.

Individual Backups: archive and rsync

You can back up and restore particular files and directories with archive tools like tar, restoring the archives later. For backups, tar is usually used with a tape device. To automatically schedule backups, you can schedule appropriate **tar** commands with the **cron** utility. The archives can be also compressed for storage savings. You can then copy the compressed archives to any medium, such as a DVD, a floppy disk, or tape. On GNOME you can use File Roller to create archives easily (Archive Manager under System Tools). The KDAT tool on KDE will back up to tapes, a front end to tar. See Chapter 10 for a discussion of compressed archives.

If you want to remote-copy a directory or files from one host to another, making a particular backup, you can use rsync, which is designed for network backups of particular directories or files, intelligently copying only those files that have been changed, rather than the contents of an entire directory. In archive mode, it can preserve the original ownership and permissions, providing corresponding users exist on the host system. The following example copies the **/home/george/myproject** directory to the **/backup** directory on the host **rabbit**, creating a corresponding **myproject** subdirectory. The **-t** specifies that this is a transfer. The remote host is referenced with an attached colon, **rabbit:**.

```
rsync -t /home/george/myproject   rabbit:/backup
```

If, instead, you want to preserve the ownership and permissions of the files, you use the **-a** (archive) option. Adding a **-z** option will compress the file. The **-v** option provides a verbose mode.

```
rsync -avz  /home/george/myproject   rabbit:/backup
```

Website	Tools
rsync.samba.org	rsync remote copy backup
amanda.org	Amanda network backup
dump.sourceforge.net	dump and restore tools
backuppc.sourceforge.net	BackupPC network or local backup using configured rsync and tar tools

TABLE 33-1 Backup Resources

A trailing slash on the source will copy the contents of the directory, rather than generating a subdirectory of that name. Here the contents of the **myproject** directory are copied to the **george-project** directory.

```
rsync -avz  /home/george/myproject/   rabbit:/backup/george-project
```

The **rsync** command is configured to use SSH remote shell by default. You can specify it or an alternate remote shell to use with the **-e** option. For secure transmission you can encrypt the copy operation with ssh. Either use the **-e ssh** option or set the **RSYNC_RSH** variable to ssh.

```
rsync -avz -e ssh  /home/george/myproject   rabbit:/backup/myproject
```

As when using **rcp**, you can copy from a remote host to the one you are on.

```
rsync -avz  lizard:/home/mark/mypics/  /pic-archive/markpics
```

You can also run rsync as a server daemon. This will allow remote users to sync copies of files on your system with versions on their own, transferring only changed files rather than entire directories. Many mirror and software FTP sites operate as rsync servers, letting you update files without having to download the full versions again. Configuration information for rsync as a server is kept in the **/etc/rsyncd.conf** file. On Linux, rsync as a server is managed through **xinetd**, using the **/etc/xinetd.d/rsync** file, which starts rsync with the **--daemon** option. In the **/etc/services** file, it is listed to run on port 873. It is off by default, but you can enable it with a tool like **chkconfig** (Fedora and SUSE) or **sysv-rc-conf** (Debian or Ubuntu).

TIP *Though it is designed for copying between hosts, you can also use rsync to make copies within your own system, usually to a directory in another partition or hard drive. In fact there are eight different ways of using rsync. Check the rsync Man page for detailed descriptions of each.*

BackupPC

BackupPC provides an easily managed local or network backup of your system or hosts on a system using configured rsync or tar tools. There is no client application to install, just configuration files. BackupPC can back up hosts on a network, including servers, or just

a single system. Data can be backed up to local hard disks or to network storage such as shared partitions or storage servers. You can configure BackupPC using your Web page configuration interface. This is the host name of your computer with the /backuppc name attached, like **http://rabbit.turtle.com/backuppc**. Detailed documentation is installed at **/usr/share/doc/BackupPC**. You can find out more about BackupPC at **backuppc.sourceforge.net**.

BackupPC uses both compression and detection of identical files to significantly reduce the size of the backup, allowing several hosts to be backed up in limited space. Once an initial backup is performed, BackupPC will only back up changed files, reducing the time of the backup significantly.

BackupPC has its own service script with which you start the BackupPC service, **/etc/init.d/backuppc**. Configuration files are located at **/etc/BackupPC**. The **config.pl** file holds BackupPC configuration options and the hosts file lists hosts to be backed up.

Amanda

To back up hosts connected to a network, you can use the Advanced Maryland Automatic Network Disk Archiver (Amanda) to archive hosts. Amanda uses tar tools to back up all hosts to a single host operating as a backup server. Backup data is sent by each host to the host operating as the Amanda server, where they are written out to a backup medium such as tape. With an Amanda server, the backup operations for all hosts become centralized in one server, instead of each host having to perform its own backup. Any host that needs to restore data simply requests it from the Amanda server, specifying the file system, date, and filenames. Backup data is copied to the server's holding disk and from there to tapes. Detailed documentation and updates are provided at **amanda.org**. For the server, be sure to install the amanda-server package, and for clients you use the amanda-clients package.

Amanda is designed for automatic backups of hosts that may have very different configurations, as well as operating systems. You can back up any host that supports GNU tools, including Mac OS X and Windows systems connected through Samba.

Amanda Commands

Amanda has its own commands corresponding to the common backup tasks, beginning with "am," such as **amdump**, **amrestore**, and **amrecover**. The commands are listed in Table 33-2. The **amdump** command is the primary backup operation.

The **amdump** command performs requested backups; it is not designed for interactive use. For an interactive backup, you use an archive tool like tar directly. The **amdump** is placed within a **cron** instruction to be run at a specified time. If, for some reason, **amdump** cannot save all its data to the backup medium (tape or disk), it will retain the data on the holding disk. The data can then later be directly written with the **amflush** command.

You can restore particular files as well as complete systems with the **amrestore** command. With the **amrecover** tool, you can select from a list of backups.

Amanda Configuration

Configuration files are placed in **/etc/amanda** and log and database files in **/var/lib/amanda**. These are created automatically when you install Amanda. You will also need to create a directory to use as a holding disk where backups are kept before writing to the tape.

Command	Description
amdump	Perform automatic backups for the file systems listed in the disklist configuration file.
amflush	Directly back up data from the holding disk to a tape.
amcleanup	Clean up if there is a system failure on the server.
amrecover	Select backups to restore using an interactive shell.
amrestore	Restore backups, either files or complete systems.
amlabel	Label the backup medium for Amanda.
amcheck	Check the backup systems and files as well as the backup tapes before backup operations.
amadmin	Back up administrative tasks.
amtape	Manage backup tapes, loading and removing them.
amverify	Check format of tapes.
amverifyrun	Check the tapes from the previous run, specify the configuration directory for the backup.
amrmtape	Remove a tape from the Amanda database, used for damaged tapes.
amstatus	Show the status of the current Amanda backup operation.

TABLE 33-2 Amanda Commands

This should be on a file system with very large available space, enough to hold the backup of your largest entire host.

/etc/amanda

Within the **/etc/amanda** directory are subdirectories for the different kind of backups you want to perform. Each directory will contain its own **amanda.conf** and **disklist** file. By default a daily backup directory is created called **DailySet1**, with a default **amanda.conf** and a sample **disklist** file. To use them, you will have to edit them to enter your system's own settings. For a different backup configuration, you can create a new directory and copy the **DailySet1 amanda.conf** and **disklist** files to it, editing them as appropriate. When you issue Amanda commands like **amdump** to perform backups, you will use the name of the **/etc/amanda** subdirectory to indicate the kind of backup you want performed.

```
amdump DailySet1
```

The **/etc/amanda** directory also contains a sample **cron** file, **crontab.sample**, that shows how a **cron** entry should look.

amanda.conf

The **amanda.conf** file contains basic configuration parameters such as the tape type and logfile, as well as holding file locations. In most cases you can just use the defaults as listed in the **DailySet1/amanda.conf** file. The file is commented in detail, telling you what entries you will have to change. You will need to set the tapedev entries to the tape device you use,

and the tape type entry for your tape drive type. In the holding disk segment, you will need to specify the partition and the directory for the holding disk you want to use. See the amanda Man page and documentation for detailed information on various options.

disklist

The **disklist** file is where you specify the file systems and partitions to be backed up. An entry lists the host, the partition, and the dump-type. The possible dump-types are defined in **amanda.conf**. The dump-types set certain parameters such as the priority of the backup and whether to use compression or not. The comp-root type will back up root partitions with compression and low priority, whereas the always-full type will back up an entire partition with no compression and the highest priority. You can define other dump-types in **amanda.conf** and use them for different partitions.

Backups will be performed in the order listed; be sure to list the more important ones first. The **disklist** file in **DailySet1** provides detailed examples.

Enabling Amanda on the Network

To use Amanda on the network, you need to run two servers on the Amanda server as well as an Amanda client on each network host. Access must be enabled for both the clients and the server.

Amanda Server

The Amanda server runs through **xinetd**, using **xinetd** service files located in **/etc/xinetd.d**. The three service files are **amanda, amidxtape,** and **amandaidx**. Then restart the **xinetd** daemon to have it take immediate effect.

For clients to be able to recover backups from the server, the clients' hostnames must be placed in the **.amandahosts** file in the server's Amanda users' directory, **/var/lib/amanda**. On the server, **/var/lib/amanda/.amandahosts** will list all the hosts that are backed up by Amanda.

Amanda Hosts

Each host needs to allow access by the Amanda server. To do this, you place the hostname of the Amanda server in each client's **.amandahosts** dot file. This file is located in the client's Amanda user home directory, **/var/lib/amanda**.

Each host needs to run the Amanda client daemon, **amanda**, which also runs under **xinetd**. On Debian, Ubuntu, and similar distributions, use the **/etc/xinetd.d/amanda** configuration file to control enabling Amanda. On Red Hat, SUSE, and similar distributions, use **chkconfig** to turn it on.

```
chkconfig amanda on
```

TIP *If your server and hosts have firewalls, you will need to allow access through the ports that Amanda uses, usually 10080, 10082, and 10083.*

Using Amanda

Backups are performed by the **amdump** command.

```
amdump DailySet1
```

An **amdump** command for each backup is placed in the Amanda **crontab** file. It is helpful to run an **amcheck** operation to make sure that a tape is ready.

```
0 16 * * 1-5 /usr/sbin/amcheck -m DailySet1
45 0 * * 2-6 /usr/sbin/amdump DailySet1
```

Before you can use a tape, you will have to label it with **amlabel**. Amanda uses the label to determine what tape should be used for a backup. Log in as the Amanda user (not root) and label the tape so that it can be used.

```
amlabel DailySet DailySet1
```

A client can recover a backup using **amrecover**. This needs to be run as the root user, not as the Amanda user. The **amrecover** command works through an interactive shell much like **ftp**, letting you list available files and select them to restore. Within the **amrecover** shell, the **ls** command will list available backups, the **add** command will select one, and the extract operation will restore it. The **lcd** command lets you change the client directory; **amrecover** will use **DailySet1** as the default, but for other configurations you will need to specify their configuration directory with the **-C** option. Should you have more than one Amanda server, you can list the one you want with the **-t** option.

```
amrecover -C DailySet1
```

To restore full system backups, you use the **amrestore** command, specifying the tape device and the hostname.

```
amrestore  /dev/rmt1  rabbit
```

To select certain files, you can pipe the output to a recovery command such as **restore** (discussed in the next section).

```
amrestore -p /dev/rmt1  rabbit mydir  | restore  -ibvf 2 -
```

Backups with dump and restore

You can back up and restore your system with the dump and restore utilities. dump can back up your entire system or perform incremental backups, saving only those files that have changed since the last backup. dump supports several options for managing the backup operation, such as specifying the size and length of storage media (see Table 33-3).

NOTE *Several disk dump tools are also available. The* ***diskdumpfmt*** *command can be used to format tapes for use by dump.* ***diskdumpctl*** *registers a dump partition with the system.* ***savecore*** *saves a* ***vmcore*** *file from the data in a dump partition. Dumped cores can be read by the* ***crash*** *tool. Check the* ***crash*** *Man page for details.*

The dump Levels

The dump utility uses *dump levels* to determine to what degree you want your system backed up. A dump level of 0 will copy file systems in their entirety. The remaining dump levels perform incremental backups, backing up only files and directories that have been

Option	Description
`-0` through `-9`	Specifies the dump level. A dump level 0 is a full backup, copying the entire file system (see also the `-h` option). Dump level numbers above 0 perform incremental backups, copying all new or modified files new in the file system since the last backup at a lower level. The default level is 9.
`-B` *records*	Lets you specify the number of blocks in a volume, overriding the end-of-media detection or length and density calculations that dump normally uses for multivolume dumps.
`-a`	Lets dump bypass any tape length calculations and write until an end-of-media indication is detected. Recommended for most modern tape drives and is the default.
`-b` *blocksize*	Lets you specify the number of kilobytes per dump record. With this option, you can create larger blocks, speeding up backups.
`-d` *density*	Specifies the density for a tape in bits per inch (default is 1,600BPI).
`-h` *level*	Files that are tagged with a user's nodump flag will not be backed up at or above this specified level. The default is 1, which will not back up the tagged files in incremental backups.
`-f` *file/device*	Backs up the file system to the specified file or device. This can be a file or tape drive. You can specify multiple filenames, separated by commas. A remote device or file can be referenced with a preceding hostname, *hostname:file.*
`-k`	Uses Kerberos authentication to talk to remote tape servers.
`-M` *file/device*	Implements a multivolume backup, where the *file* written to is treated as a prefix and the suffix consisting of a numbered sequence from 001 is used for each succeeding file, *file*001, *file*002, and so on. Useful when backup files need to be greater than the Linux **ext3** 2 GB file size limit.
`-n`	Notifies operators if a backup needs operator attention.
`-s` *feet*	Specifies the length of a tape in feet. dump will prompt for a new tape when the length is reached.
`-S`	Estimates the amount of space needed to perform a backup.
`-T` *date*	Allows you to specify your own date instead of using the **/etc/dumpdates** file.
`-u`	Writes an entry for a successful update in the **/etc/dumpdates** file.
`-W`	Detects and displays the file systems that need to be backed up. This information is taken from the **/etc/dumpdates** and **/etc/fstab** files.
`-w`	Detects and displays the file systems that need to be backed up, drawing only on information in **/etc/fstab**.

TABLE 33-3 Options for dump

created or modified since the last lower-level backup. A dump level of 1 will back up only files that have changed since the last level 0 backup. The dump level 2, in turn, will back up only files that have changed since the last level 1 backup (or 0 if there is no level 1), and so on up to dump level 9. You can run an initial complete backup at dump level 0 to back up your entire system and then run incremental backups at certain later dates, backing up only the changes since the full backup.

Using dump levels, you can devise certain strategies for backing up a file system. It is important to keep in mind that an incremental backup is run on changes from the last lower-level backup. For example, if the last backup was 6 and the next backup was 8, then the level 8 will back up everything from the level 6 backup. The sequence of the backups is important. If there were three backups with levels 3, then 6, and then 5, the level 5 backup would take everything from the level 3 backup, not stopping at level 6. Level 3 is the next-*lower*-level backup for level 5, in this case. This can make for some complex incremental backup strategies. For example, if you want each succeeding incremental backup to include all the changes from the preceding incremental backups, you can run the backups in descending dump level order. Given a backup sequence of 7, 6, and 5, with 0 as the initial full backup, 6 would include all the changes to 7, because its next lower level is 0. Then 5 would include all the changes for 7 and 6, also because its next lower level is 0, making all the changes since the level 0 full backup. A simpler way to implement this is to make the incremental levels all the same. Given an initial level of 0, and then two backups both with level 1, the last level 1 would include all the changes from the backup with level 0, since level 0 is the next *lower* level—not the previous level 1 backup.

Recording Backups

Backups are recorded in the **/etc/dumpdates** file. This file will list all the previous backups, specifying the file system they were performed on, the dates they were performed, and the dump level used. You can use this information to restore files from a specified backup. Recall that the **/etc/fstab** file records the dump level as well as the recommended backup frequency for each file system. With the **-W** option, dump will analyze both the **/etc/dumpdates** and **/etc/fstab** files to determine which file systems need to be backed up. The **dump** command with the **-w** option just uses **/etc/fstab** to report the file systems ready for backup.

Operations with dump

The **dump** command takes as its arguments the dump level, the device it is storing the backup on, and the device name of the file system that is being backed up. If the storage medium (such as a tape) is too small to accommodate the backup, **dump** will pause and let you insert another. **dump** supports backups on multiple volumes. The **u** option will record the backup in the **/etc/dumpdates** file. In the following example, an entire backup (dump level 0) is performed on the file system on the **/dev/hda3** hard disk partition. The backup is stored on a tape device, **/dev/tape**.

```
dump -0u -f /dev/tape /dev/hda5
```

NOTE *You can use the* **mt** *command to control your tape device;* **mt** *has options to rewind, erase, and position the tape. The* **rmt** *command controls a remote tape device.*

The storage device can be another hard disk partition, but it is usually a tape device. When you installed your system, your system most likely detected the device and set up **/dev/tape** as a link to it (just as it did with your CD-ROMs). If the link was not set up, you have to create it yourself or use the device name directly. Tape devices can have different device names, depending on the model or interface. SCSI tape devices are labeled with the prefix **st**, with a number attached for the particular device: **st0** is the first SCSI tape device. To use it in the **dump** command, just specify its name.

```
dump -0u -f /dev/st0 /dev/hda5
```

Should you need to back up to a device located on another system on your network, you have to specify that hostname for the system and the name of its device. The hostname is entered before the device name and delimited with a colon. In the following example, the user backs up file system **/dev/hda5** to the SCSI tape device with the name **/dev/st1** on the **rabbit.mytrek.com** system:

```
dump -0u -f rabbit.mytrek.com:/dev/st0 /dev/hda5
```

The **dump** command works on one file system at a time. If your system has more than one file system, you will need to issue a separate **dump** command for each.

TIP *You can use the system* ***cron*** *utility to schedule backups using dump at specified times.*

Recovering Backups

You use the **restore** command either to restore an entire file system or to just retrieve particular files. **restore** will extract files or directories from a backup archive and copy them to the current working directory. Make sure you are in the directory you want the files restored to when you run **restore**. **restore** will also generate any subdirectories as needed, and it has several options for managing the restore operation (see Table 33-4).

To recover individual files and directories, you run **restore** in an interactive mode using the **-i** option. This will generate a shell with all the directories and files on the tape, letting you select the ones you want to restore. When you are finished, **restore** will then retrieve from a backup only those files you selected. This shell has its own set of commands that you can use to select and extract files and directories (see Table 33-5). The following command will generate an interactive interface listing all the directories and files backed up on the tape in the **/dev/tape** device:

```
restore -ivf /dev/tape
```

This command will generate a shell encompassing the entire directory structure of the backup. You are given a shell prompt and can use the **cd** command to move to different directories and the **ls** command to list files and subdirectories. You use the **add** command to tag a file or directory for extraction. Should you later decide not to extract it, you can use the **delete** command to remove it from the tagged list. Once you have selected all the items you want, you enter the **extract** command to retrieve them from the backup archive. To quit the restore shell, you enter **quit**. The **help** command will list the restore shell commands.

Operation	Description
-C	Lets you check a backup by comparing files on a file system with those in a backup.
-i	The interactive mode for restoring particular files and directories in a backup. A shell interface is generated where the user can use commands to specify file and directories to restore (see Table 33-5).
-R	Instructs **restore** to request a tape that is part of a multivolume backup, from which to continue the restore operation. Helpful when multivolume restore operations are interrupted.
-r	Restores a file system. Make sure that a newly formatted partition has been mounted and that you have changed to its top directory.
-t	Lists the contents of a backup or specified files in it.
-x	Extracts specified files or directories from a backup. A directory is restored with all its subdirectories. If no file or directory is specified, the entire file system is restored.
Additional Option	**Description**
-b *blocksize*	Specifies a block size; otherwise, **restore** will dynamically determine it from the block device.
-f *file/device*	Restores the backup on the specified file or device. Specify a hostname for remote devices.
-F *script*	Runs a script at the beginning of the restore.
-k	Uses Kerberos authentication for remote devices.
-h	Extracts only the specified directories, without their subdirectories.
-M *file/device*	Restores from multivolume backups, where *file* is treated as a prefix and the suffix is a numbered sequence, *file*001, *file*002.
-N	Displays the names of files and directories, does not extract them.
-T *directory*	Specifies a directory to use for the storage of temporary files. The default value is **/tmp**.
-v	The verbose mode, where each file and its file type that **restore** operates on is displayed.
-y	By default, **restore** will query the operator to continue if an error occurs, such as bad blocks. This option suppresses that query, allowing **restore** to automatically continue.

TABLE 33-4 Operations and Options for **restore**

If you need to restore an entire file system, use **restore** with the **-r** option. You can restore the file system to any blank formatted hard disk partition of adequate size, including the file system's original partition. It may be advisable, if possible, to restore the file system on another partition and check the results.

Command	Description
add [*arg*]	Adds files or directories to the list of files to be extracted. Such tagged files display an * before their names when listed with **ls**. All subdirectories of a tagged directory are also extracted.
cd *arg*	Changes the current working directory.
delete [*arg*]	Deletes a file or directory from the extraction list. All subdirectories for deleted directories will also be removed.
extract	Extracts files and directories on the extraction list.
help	Displays a list of available commands.
ls [*arg*]	Lists the contents of the current working directory or a specified directory.
pwd	Displays the full pathname of the current working directory.
quit	Exits the restore interactive mode shell. The **quit** command does not perform any extraction, even if the extraction list still has items in it.
setmodes	Sets the owner, modes, and times for all files and directories in the extraction list. Used to clean up an interrupted restore.
verbose	In the verbose mode, each file is listed as it is extracted. Also, the **ls** command lists the inode numbers for files and directories.

TABLE 33-5 Interactive Mode Shell Commands for **restore**

Restoring an entire file system involves setting up a formatted partition, mounting it to your system, and then changing to its top directory to run the **restore** command. First you should use **mkfs** to format the partition where you are restoring the file system, and then mount it onto your system. Then you use **restore** with the **-r** option and the **-f** option to specify the device holding the file system's backup. In the next example, the user formats and mounts the **/dev/hda5** partition and then restores on that partition the file system backup, currently on a tape in the **/dev/tape** device.

```
mkfs /dev/hda5
mount /dev/hda5 /mystuff
cd /mystuff
restore -rf /dev/tape
```

To restore from a backup device located on another system on your network, you have to specify that hostname for the system and the name of its device. The hostname is entered before the device name and delimited with a colon. In the following example, the user restores a file system from the backup on the tape device with the name **/dev/tape** on the **rabbit.mytrek.com** system:

```
restore -rf rabbit.mytrek.com:/dev/tape
```

This page intentionally left blank

PART VIII

Network Administration Services

CHAPTER 34
Administering TCP/IP Networks

CHAPTER 35
Network Autoconfiguration with IPv6, DHCPv6, and DHCP

CHAPTER 36
NFS and NIS

CHAPTER 37
Distributed Network File Systems

This page intentionally left blank

34 CHAPTER

Administering TCP/IP Networks

Linux systems are configured to connect into networks that use the TCP/IP protocols. These are the same protocols that the Internet uses, as do many local area networks (LANs). TCP/IP is a robust set of protocols designed to provide communications among systems with different operating systems and hardware. The TCP/IP protocols were developed in the 1970s as a special DARPA project to enhance communications between universities and research centers. These protocols were originally developed on Unix systems, with much of the research carried out at the University of California, Berkeley. Linux, as a version of Unix, benefits from much of this original focus on Unix. Currently, TCP/IP protocol development is managed by the Internet Engineering Task Force (IETF), which, in turn, is supervised by the Internet Society (ISOC). The ISOC oversees several groups responsible for different areas of Internet development, such as the Internet Assigned Numbers Authority (IANA), which is responsible for Internet addressing (see Table 34-1). Over the years, TCP/IP protocol standards and documentation have been issued in the form of Request for Comments (RFC) documents. Check the most recent ones for current developments at the IETF website at **ietf.org**.

TCP/IP Protocol Suite

The TCP/IP protocol suite actually consists of different protocols, each designed for a specific task in a TCP/IP network. The three basic protocols are the Transmission Control Protocol (TCP), which handles receiving and sending out communications; the Internet Protocol (IP), which handles the actual transmissions; and the User Datagram Protocol (UDP), which also handles receiving and sending packets. The IP protocol, which is the base protocol that all others use, handles the actual transmissions, the packets of data with sender and receiver information in each. The TCP protocol is designed to work with cohesive messages or data. This protocol checks received packets and sorts them into their designated order, forming the original message. For data sent out, the TCP protocol breaks the data into separate packets, designating their order. The UDP protocol, meant to work on a much more raw level, also breaks down data into packets but does not check their order. The TCP/IP protocol is designed to provide stable and reliable connections that ensure that

Group	Title	Description
ISOC	Internet Society	Professional membership organization of Internet experts that oversees boards and task forces dealing with network policy issues **isoc.org**
IESG	The Internet Engineering Steering Group	Responsible for technical management of IETF activities and the Internet standards process **ietf.org/iesg.html**
IANA	Internet Assigned Numbers Authority	Responsible for Internet Protocol (IP) addresses **iana.org**
IAB	Internet Architecture Board	Defines the overall architecture of the Internet, providing guidance and broad direction to the IETF **iab.org**
IETF	Internet Engineering Task Force	Protocol engineering and development arm of the Internet **ietf.org**

TABLE 34-1 TCP/IP Protocol Development Groups

all data is received and reorganized into its original order. UDP, on the other hand, is designed to simply send as much data as possible, with no guarantee that packets will all be received or placed in the proper order. UDP is often used for transmitting very large amounts of data of the type that can survive the loss of a few packets—for example, temporary images, video, and banners displayed on the Internet.

Other protocols provide various network and user services. The Domain Name Service (DNS) provides address resolution. The File Transfer Protocol (FTP) provides file transmission, and the Network File System (NFS) provides access to remote file systems. Table 34-2 lists the different protocols in the TCP/IP protocol suite. These protocols make use of either the TCP or UDP protocol to send and receive packets, which, in turn, uses the IP protocol for actually transmitting the packets.

In a TCP/IP network, messages are broken into small components, called *datagrams*, which are then transmitted through various interlocking routes and delivered to their destination computers. Once received, the datagrams are reassembled into the original message. Datagrams themselves can be broken down into smaller packets. The *packet* is the physical message unit actually transmitted among networks. Sending messages as small components has proved to be far more reliable and faster than sending them as one large, bulky transmission. With small components, if one is lost or damaged, only that component must be resent, whereas if any part of a large transmission is corrupted or lost, the entire message has to be resent.

The configuration of a TCP/IP network on your Linux system is implemented using a set of network configuration files (Table 34-6, later in this chapter, provides a listing). Many of these can be managed using administrative programs provided by your distribution on your root user desktop. You can also use the more specialized programs, such as netstat, ifconfig, Wireshark, and route. Some configuration files are easy to modify using a text editor.

TCP/IP networks are configured and managed with a set of utilities: ifconfig, route, and netstat. The ifconfig utility operates from your root user desktop and enables you to configure

your network interfaces fully, adding new ones and modifying others. The ifconfig and route utilities are lower-level programs that require more specific knowledge of your network to use effectively. The netstat utility provides you with information about the status of your network connections. Wireshark is a network protocol analyzer that lets you capture packets as they are transmitted across your network, selecting those you want to check.

Transport	Description
TCP	Transmission Control Protocol; places systems in direct communication
UDP	User Datagram Protocol
IP	Internet Protocol; transmits data
ICMP	Internet Control Message Protocol; status messages for IP
Routing	**Description**
RIP	Routing Information Protocol; determines routing
OSPF	Open Shortest Path First; determines routing
Network Address	**Description**
ARP	Address Resolution Protocol; determines unique IP address of systems
DNS	Domain Name Service; translates hostnames into IP addresses
RARP	Reverse Address Resolution Protocol; determines addresses of systems
User Service	**Description**
FTP	File Transfer Protocol; transmits files from one system to another using TCP
TFTP	Trivial File Transfer Protocol; transfers files using UDP
Telnet	Remote login to another system on the network
SMTP	Simple Mail Transfer Protocol; transfers email between systems
RPC	Remote Procedure Call; allows programs on remote systems to communicate
Gateway	**Description**
EGP	Exterior Gateway Protocol; provides routing for external networks
GGP	Gateway-to-Gateway Protocol; provides routing between Internet gateways
IGP	Interior Gateway Protocol; provides routing for internal networks
Network Service	**Description**
NFS	Network File System; allows mounting of file systems on remote machines
NIS	Network Information Service; maintains user accounts across a network
BOOTP	Boot Protocol; starts system using boot information on server for network
SNMP	Simple Network Management Protocol; provides status messages on TCP/IP configuration
DHCP	Dynamic Host Configuration Protocol; automatically provides network configuration information to host systems

TABLE 34-2 TCP/IP Protocol Suite

Configuring Networks on GNOME and KDE

Both GNOME and KDE provide network configuration tools for setting up your network connections easily. On GNOME, the **network-admin** tool displays a Network Settings window with panels for Connections, General, Hosts, and DNS. The Connections entry lists possible connections like Wireless, Hardwire, and Modem. Active connections will have check marks in their check boxes. You can deactivate a connection by unchecking its entry. To configure a connection, select it and click Properties. This opens a Settings window for your connected interface. Here you can select the kind of configuration you want, like DHCP, Static IP, or zeroconf. If you select Static IP you can enter the IP address, Netmask, and Gateway address.

The General panel lets you set the hostname and domain name. On the DNS panel you can specify DNS servers and search domains (on a DHCP configuration these will be specified automatically). The Hosts panel lists all the manually specified host names on your network. You can add or delete entries.

If you have made changes, you can save them by clicking the disk button on the main window. You will be asked to specify a location or add a new one (the current one will be listed by default). You can set up different configurations for various locations like office or home. Then load the location you want from the Locations pop-up menu.

On KDE, you select Network Setting in the Internet & Network menu of the Control Center. Click the Administrative Mode button to let you change entries. Four panels are displayed for Network Interfaces, Routes, Domain Name System, and Network Profiles. On the Network Interfaces panel, your connected network interfaces are listed. There are buttons to Enable or Disable an interface. To configure a device, click on its entry and click the Configure Interface button. This opens a window where you can seleclt Automatic or Manual configurations. For automatic configuration you can use DHCP (the default). Manual configuration lets you enter an IP address and Netmask. The Routes panel is where you enter the Gateway address (on DHCP configurations, this will be already entered.). The domain name system panel lets you specify the host name of your computer, your domain name, and any domain name servers for your network. You can also set up static hosts on your network. The Network Profiles panel lets you save or create a network profile such as one for home or office.

Zero Configuration Networking (zeroconf): Avahi and Link Local Addressing

Zero Configuration Networking (zeroconf) allows the setup of non-routable private networks without the need of a DHCP server or static IP addresses. A zeroconf configuration lets users automatically connect to a network and access all network resources like printers, without having to perform any configuration. On Linux, zeroconf networking is implemented by Avahi (**avahi.org**). Avahi includes multicast DNS (mDNS) and DNS service discovery (DNS-SD) support which automatically detects services on a network. IP addresses are determined using either IPv6 or IPv4 Link Local (IPv4LL) addressing. IPv4 Link Local addresses are assigned from the 168.254.0.0 network pool. Derived from Apple's Bonjour zeroconf implementation, it is a free and open source version currently used by desktop tools like the GNOME virtual file system. Some distributions implement full zeroconf network support (Debian and Ubuntu),

while others make use the the DNS service discovery (Red Hat and Fedora). Many distributions include the KDE zeroconf solution using Avahi (kdnssd). This is located in the kdnnsd-avahi packages. Use the KDE Control Center Service Discovery panel (Internet & Network) to specify your domain. Then enter **zeorconf:/** in a KDE file manager window.

IPv4 and IPv6

Traditionally, a TCP/IP address is organized into four segments, consisting of numbers separated by periods. This is called the *IP address*. The IP address actually represents a 32-bit integer whose binary values identify the network and host. This form of IP addressing adheres to Internet Protocol, version 4, also known as IPv4. IPv4, the kind of IP addressing described here, is still in wide use.

Currently, a new version of the IP protocol called Internet Protocol, version 6 (IPv6) is gradually replacing the older IPv4 version. IPv6 expands the number of possible IP addresses by using 128 bits. It is fully compatible with systems still using IPv4. IPv6 addresses are represented differently, using a set of eight 16-bit segments, each separated from the next by a colon. Each segment is represented by a hexadecimal number. A sample address is

```
FEC0:0:0:0:800:BA98:7654:3210
```

Advantages for IPv6 include the following:

- It features simplified headers that allow for faster processing.
- It provides support for encryption and authentication along with virtual private networks (VPN) using the integrated IPsec protocol.
- One of its most significant advantages lies in its extending the address space to cover 2 to the power of 128 possible hosts (billions of billions). This extends far beyond the 4.2 billion supported by IPv4.
- It supports stateless autoconfiguration of addresses for hosts, bypassing the need for DHCP to configure such addresses. Addresses can be generated directly using the MAC (Media Access Control) hardware address of an interface.
- It provides support for Quality of Service (QoS) operations, providing sufficient response times for services like multimedia and telecom tasks.
- Multicast capabilities are built into the protocol, providing direct support for multimedia tasks. Multicast addressing also provides that same function as IPv4 broadcast addressing.
- More robust transmissions can be ensured with anycast addressing, where packets can be directed to an anycast group of systems, only one of which needs to receive them. Multiple DNS servers supporting a given network can be designated as an anycast group, of which only one DNS server needs to receive the transmission, providing greater likelihood that the transmission will go through.
- It provides better access for mobile nodes such as PDAs, notebooks, and cell phones.

TCP/IP Network Addresses

As noted previously, the traditional IPv4 TCP/IP address is organized into four segments, consisting of numbers separated by periods. This kind of address is still in widespread use and is what people commonly refer to as an *IP address*. Part of an IP address is used for the network address, and the other part is used to identify a particular interface on a host in that network. You should realize that IP addresses are assigned to interfaces—such as Ethernet cards or modems—and not to the host computer. Usually a computer has only one interface and is accessed using only that interface's IP address. In that regard, an IP address can be thought of as identifying a particular host system on a network, and so the IP address is usually referred to as the *host address*.

In fact, though, a host system can have several interfaces, each with its own IP address. This is the case for computers that operate as gateways and firewalls from the local network to the Internet. One interface usually connects to the LAN and another to the Internet, as by two Ethernet cards. Each interface (such as an Ethernet card) has its own IP address. If you use a modem to connect to an ISP, you must set up a PPP interface that also has its own IP address (usually dynamically assigned by the ISP). Remembering this distinction is important if you plan to use Linux to set up a local or home network, using Linux as your gateway machine to the Internet (see the section "IP Masquerading" in Chapter 20).

IPv4 Network Addresses

The IP address is divided into two parts: one part identifies the network, and the other part identifies a particular host. The network address identifies the network of which a particular interface on a host is a part. Two methods exist for implementing the network and host parts of an IP address: the original class-based IP addressing and the current Classless Interdomain Routing (CIDR) addressing. Class-based IP addressing designates officially predetermined parts of the address for the network and host addresses, whereas CIDR addressing allows the parts to be determined dynamically using a netmask.

Class-Based IP Addressing

Originally, IP addresses were organized according to classes. On the Internet, networks are organized into three classes depending on their size—classes A, B, and C. A class A network uses only the first segment for the network address and the remaining three for the host, allowing a great many computers to be connected to the same network. Most IP addresses reference smaller, class C, networks. For a class C network, the first three segments are used to identify the network, and only the last segment identifies the host. Altogether, this forms a unique address with which to identify any network interface on computers in a TCP/IP network. For example, in the IP address 192.168.1.72, the network part is 192.168.1 and the interface/host part is 72. The interface/host is a part of a network whose own address is 192.168.1.0.

In a class C network, the first three numbers identify the network part of the IP address. This part is divided into three network numbers, each identifying a subnet. Networks on the Internet are organized into subnets, beginning with the largest and narrowing to small subnetworks. The last number is used to identify a particular computer, referred to as a *host*. You can think of the Internet as a series of networks with subnetworks; these subnetworks have their own subnetworks. The rightmost number identifies the host computer, and the

number preceding it identifies the subnetwork of which the computer is a part. The number to the left of that identifies the network the subnetwork is part of, and so on. The Internet address 192.168.187.4 references the fourth computer connected to the network identified by the number 187. Network 187 is a subnet to a larger network identified as 168. This larger network is itself a subnet of the network identified as 192. Here's how it breaks down:

192.168.187.4	IPv4 address
192.168.187	Network identification
4	Host identification

Netmask

Systems derive the network address from the host address using the netmask. You can think of an IP address as a series of 32 binary bits, some of which are used for the network and the remainder for the host. The *netmask* has the network set of bits set to 1s, with the host bits set to 0s (see Figure 34-1). In a netmask for a standard class-based IP address, all the numbers in the network part of the host address are set to 255, and the host part is set to 0. This has the effect of setting all the binary bits making up the network address in the netmask to 1s. This, then, is your netmask. So, the netmask for the Class C host address 192.168.1.72 is 255.255.255.0. The network part, 192.168.1, has been set to 255.255.255, and the host part, 72, has been set to 0. Systems can then use your netmask to derive your network address from your host address. They can determine what part of your host address makes up your network address and what those numbers are.

For those familiar with computer programming, a bitwise AND operation on the netmask and the host address results in zeroing the host part, leaving you with the network part of the host address. You can think of the address as being implemented as a four-byte integer, with each byte corresponding to a segment of the address. In a class C address, the three network segments correspond to the first three bytes and the host segment corresponds to the fourth byte. A netmask is designed to mask out the host part of the address, leaving the network segments alone. In the netmask for a standard class C network, the first three bytes are all 1s and the last byte consists of 0s. The 0s in the last byte mask out the host part of the address, and the 1s in the first three bytes leave the network part of the address alone. Figure 34-1 shows the bitwise operation of the netmask on the address 192.168.1.4. When it is applied to the address 192.168.1.4, the network address (192.168.1) remains and the host address (4) is masked out, giving you 192.168.1.0 as the network address.

The netmask as used in Classless Interdomain Routing (CIDR) is much more flexible. Instead of having the size of the network address and its mask determined by the network class, it is determined by a number attached to the end of the IP address. This number simply specifies the size of the network address, how many bits in the address it takes up. For example, in an IP address whose network part takes up the first three bytes (segments), the number of bits used for that network part is 24—eight bits to a byte (segment). Instead of using a netmask to determine the network address, the number for the network size is attached to the end of the address with a slash, as shown here:

```
192.168.1.72/24
```

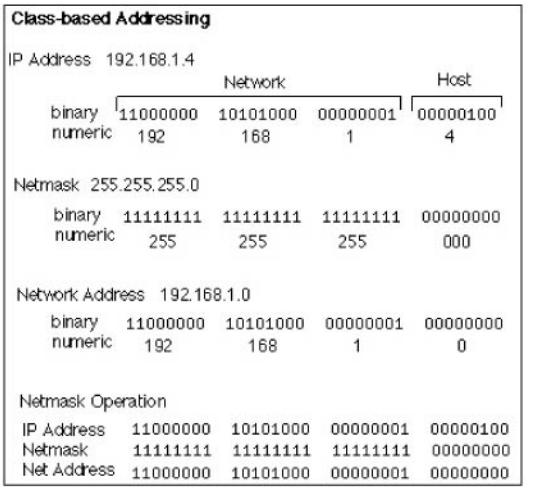


FIGURE 34-1 Class-based netmask operations

CIDR gives you the advantage of specifying networks that are any size in bits, instead of only three possible segments. You could have a network whose address takes up 14 bits, 22 bits, or even 25 bits. The host address can use whatever bits are left over. An IP address with 21 bits for the network can cover host addresses using the remaining 11 bits, 0 to 2047.

Classless Interdomain Routing (CIDR)

Currently, the class-based organization of IP addresses is being replaced by the CIDR format. CIDR was designed for midsized networks, those between a class C and classes with numbers of hosts greater than 256 and smaller than 65,534. A class C network–based IP address uses only one segment for hosts, an 8-bit integer with a maximum value of 256, for hosts. A class B network–based IP address uses two segments, which make up a 16-bit integer whose maximum value is 65,534. You can think of an address as a 32-bit integer taking up four bytes, where each byte is 8 bits. Each segment conforms to one of the four bytes. A class C network uses three segments, or 24 bits, to make up its network address. A class B network, in turn, uses two segments, or 16 bits, for its address. With this scheme, allowable host and network addresses are changed an entire byte at a time, segment to segment. With CIDR addressing, you can define host and network addresses by bits, instead of whole segments. For example, you can use CIDR addressing to expand the host segment from 8 bits to 9, rather than having to jump it to a class B 16 bits (two segments).

CIDR addressing notation achieves this by incorporating netmask information in the IP address (the netmask is applied to an IP address to determine the network part of the address). In the CIDR notation, the number of bits making up the network address is placed after the IP address, following a slash. For example, the CIDR form of the class C 192.168.187.4 IP address is

```
192.168.187.4/24
```

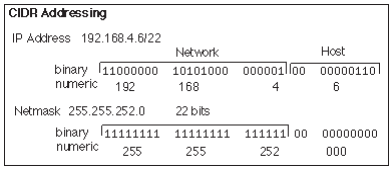


FIGURE 34-2 CIDR addressing

Figure 34-2 shows an example of a CIDR address and its network mask. The IP address is 192.168.1.6 with a network mask of 22 bits, 192.168.1.6/22. The network address takes up the first 22 bits of the IP address, and the remaining 10 bits are used for the host address. The host address takes up the equivalent of a class-based IP address's fourth segment (8 bits) and 2 bits from the third segment.

Table 34-3 lists the different IPv4 CIDR network masks available along with the maximum number of hosts. Both the short forms and the full forms of the netmasks are listed.

IPv4 CIDR Addressing

The network address for any standard class C IPv4 address takes up the first three segments, 24 bits. If you want to create a network with a maximum of 512 hosts, you can give them IP addresses where the network address is 23 bits and the host address takes up 9 bits (0–511). The IP address notation remains the same, however, using the four 8-bit segments. This means a given segment's number could be used for both a network address

Short Form	Full Form	Maximum Number of Hosts
/8	/255.0.0.0	16,777,215 (A class)
/16	/255.255.0.0	65,534 (B class)
/17	/255.255.128.0	32,767
/18	/255.255.192.0	16,383
/19	/255.255.224.0	8191
/20	/255.255.240.0	4095
/21	/255.255.248.0	2047
/22	/255.255.252.0	1023
/23	/255.255.254.0	511
/24	/255.255.255.0	255 (C class)
/25	/255.255.255.128	127
/26	/255.255.255.192	63
/27	/255.255.255.224	31
/28	/255.255.255.240	15
/29	/255.255.255.248	7
/30	/255.255.255.252	3

TABLE 34-3 CIDR IPv4 Network Masks

and a host address. Segments are no longer wholly part of either the host address or the network address. Assigning a 23-bit network address and a 9-bit host address means that the number in the third segment is part of both the network address and the host address, the first 7 bits for the network and the last bit for the host. In this following example, the third number, 145, is used as the end of the network address and as the beginning of the host address:

```
192.168.145.67/23
```

This situation complicates CIDR addressing, and in some cases the only way to represent the address is to specify two or more network addresses. Check RFC 1520 at **ietf.org** for more details.

NOTE *A simple way to calculate the number of hosts a network can address is to take the number of bits in its host segment as a power of 2, and then subtract 2—that is, 2 to the number of host bits, minus 2. For example, an 8-bit host segment would be 2 to the power of 8, which equals 256. Subtract 2 (1 for the broadcast address, 255, and 1 for the zero value, 000) to leave you with 254 possible hosts.*

CIDR also allows a network administrator to take what is officially the host part of an IP address and break it up into subnetworks with fewer hosts. This is referred to as *subnetting.* A given network will have its official IP network address recognized on the Internet or by a larger network. The network administrator for that network could, in turn, create several smaller networks within it using CIDR network masking. A classic example is to take a standard class C network with 254 hosts and break it up into two smaller networks, each with 64 hosts. You do this by using a CIDR netmask to take a bit from the host part of the IP address and use it for the subnetworks. Numbers within the range of the original 254 addresses whose first bit would be set to 1 would represent one subnet, and the others, whose first bit would be set to 0, would constitute the remaining network. In the network whose network address is 192.168.187.0, where the last segment is used for the hostnames, that last host segment could be further split into two subnets, each with its own hosts. For two subnets, you would use the first bit in the last 8-bit segment for the network. The remaining 7 bits could then be used for host addresses, giving you a range of 127 hosts per network. The subnet whose bit is set to 0 would have a range of 1 to 127, with a CIDR netmask of 25. The 8-bit segment for the first host would be 00000001. So the host with the address of 1 in that network would have this IP address:

```
192.168.187.1/25
```

For the subnet where the first bit is 1, the first host would have an address of 129, with the CIDR netmask of 25, as shown here. The 8-bit sequence for the first host would be 10000001.

```
192.168.187.129/25
```

Each subnet would have a set of 126 addresses, the first from 1 to 126, and the second from 129 to 254; 127 is the broadcast address for the first subnet, and 128 is the network

address for the second subnet. The possible subnets and their masks that you could use are shown here:

Subnetwork	CIDR Address	Binary Mask
First subnet network address	.0/25	00000000
Second subnet network address	.128/25	10000000
First subnet broadcast address	.127/25	01111111
Second subnet broadcast address	.255/25	11111111
First address in first subnet	.1/25	00000001
First address in second subnet	.129/25	10000001
Last address in first subnet	.126/25	01111110
Last address in second subnet	.254/25	11111110

IPv6 CIDR Addressing

IPv6 CIDR addressing works much the same as with the IPv4 method. The number of bits used for the network information is indicated by a number following the address. A host (interface) address could take up much more than the 64 bits that it usually does in an IPv6 address, making the network prefix (address) section smaller than 64 bits. How many bits that the network prefix uses is indicated by the following number. In the next example the network prefix (address) uses only the first 48 bits of the IPv6 address, and the host address uses the remaining 80 bits:

```
FEC0:0000:0000:0000:FEDC:BA98:7654:3210/48
```

You can also use a two-colon notation (::) for the compressed version:

```
FEC0::FEDC:BA98:7654:3210/48
```

Though you can use CIDR to subnet addresses, IPv6 also supports a subnet field that can be used for subnets.

Obtaining an IP Address

IP addresses are officially allocated by IANA, which manages all aspects of Internet addressing (**iana.org**). IANA oversees Internet Registries (IRs), which, in turn, maintain Internet addresses on regional and local levels. The Internet Registry for the Americas is the American Registry for Internet Numbers (ARIN), whose website is at **arin.net**. These addresses are provided to users by Internet service providers (ISPs). You can obtain your own Internet address from an ISP, or if you are on a network already connected to the Internet, your network administrator can assign you one. If you are using an ISP, the ISP may temporarily assign one from a pool it has on hand with each use.

IPv4 Reserved Addresses

Certain numbers are reserved. The numbers 127, 0, or 255 cannot be part of an official IP address. The number 127 is used to designate the network address for the loopback interface on your system. The loopback interface enables users on your system to communicate with

each other within the system without having to route through a network connection. Its network address would be 127.0.0.0, and its IP address is 127.0.0.1. For class-based IP addressing, the number 255 is a special broadcast identifier you can use to broadcast messages to all sites on a network. Using 255 for any part of the IP address references all nodes connected at that level. For example, 192.168.255.255 broadcasts a message to all computers on network 192.168, all its subnetworks, and their hosts. The address 192.168.187.255 broadcasts to every computer on the local network. If you use 0 for the network part of the address, the host number references a computer within your local network. For example, 0.0.0.6 references the sixth computer in your local network. If you want to broadcast to all computers on your local network, you can use the number 0.0.0.255. For CIDR IP addressing, the broadcast address may appear much like a normal IP address. As indicated in the preceding section, CIDR addressing allows the use of any number of bits to make up the IP address for either the network or the host part. For a broadcast address, the host part must have all its bits set to 1.

A special set of numbers is reserved for use on non-Internet LANs (RFC 1918). These are numbers that begin with the special network number 192.168 (for class C networks), as used in these examples. If you are setting up a LAN, such as a small business or a home network, you are free to use these numbers for your local machines. You can set up an intranet using network cards, such as Ethernet cards and Ethernet hubs, and then configure your machines with IP addresses starting from 192.168.1.1. The host segment can go up to 256. If you have three machines on your home network, you could give them the addresses 192.168.1.1, 192.168.1.2, and 192.168.1.3. You can implement Internet services, such as FTP, web, and mail services, on your local machines and use any of the Internet tools to make use of those services. They all use the same TCP/IP protocols used on the Internet. For example, with FTP tools, you can transfer files among the machines on your network. With mail tools, you can send messages from one machine to another, and with a web browser, you can access local websites that may be installed on a machine running its own web servers. If you want to have one of your machines connected to the Internet or some other network, you can set it up to be a gateway machine. By convention, the gateway machine is usually given the address 192.168.1.1. With a method called *IP masquerading*, you can have any of the non-Internet machines use a gateway to connect to the Internet.

Numbers are also reserved for class A and class B non-Internet local networks. Table 34-4 lists these addresses. The possible addresses available span from 0 to 255 in the host segment of the address. For example, class B network addresses range from 172.16.0.0 to 172.31.255.255, giving you a total of 32,356 possible hosts. The class C network ranges from 192.168.0.0 to 192.168.255.255, giving you 256 possible subnetworks, each with 256 possible hosts. The network address 127.0.0.0 is reserved for a system's loopback interface, which allows it to communicate with itself, enabling users on the same system to send messages to each other.

IPv4 Private Network Addresses	Network Classes
10.0.0.0	Class A network
172.16.0.0–172.31.255.255	Class B network
192.168.0.0	Class C network
127.0.0.0	Loopback network (for system self-communication)

Table 34-4 Non-Internet IPv4 Local Network IP Addresses

Broadcast Addresses

The broadcast address allows a system to send the same message to all systems on your network at once. With IPv4 class-based IP addressing, you can easily determine the broadcast address using your host address: the broadcast address has the host part of your address set to 255. The network part remains untouched. So the broadcast address for the host address 192.168.1.72 is 192.168.1.255 (you combine the network part of the address with 255 in the host part). For CIDR IP addressing, you need to know the number of bits in the netmask. The remaining bits are set to 1 (see Figure 34-3). For example, an IP address of 192.168.4.6/22 has a broadcast address of 192.168.7.255/22. In this case, the first 22 bits are the network address and the last 10 bits are the host part set to the broadcast value (all 1s).

In fact, you can think of a class C broadcast address as merely a CIDR address using 24 bits (the first three segments) for the network address, and the last 8 bits (the fourth segment) as the broadcast address. The value 255 expressed in binary terms is simply 8 bits that are all 1s. 255 is the same as 11111111.

IP Address	Broadcast Address	IP Broadcast Number	Binary Equivalent
192.168.1.72	192.168.1.255	255	11111111
192.168.4.6/22	192.168.7.255/22	7.255 (last 2 bits in 7)	1111111111

Gateway Addresses

Some networks have a computer designated as the gateway to other networks. Every connection to and from a network to other networks passes through this gateway computer. Most local networks use gateways to establish a connection to the Internet. If you are on this type of network, you must provide the gateway address. If your network does not have a connection to the Internet, or a larger network, you may not need a gateway address. The gateway address is the address of the host system providing the gateway service to the network. On many networks, this host is given a host ID of 1: the gateway address for a network with the address 192.168.1 would be 192.168.1.1, but this is only a convention. To be sure of your gateway address, ask your network administrator.

Name Server Addresses

Many networks, including the Internet, have computers that provide a Domain Name Service (DNS) that translates the domain names of networks and hosts into IP addresses. These are known as the network's *domain name servers.* The DNS makes your computer identifiable on a network, using only your domain name, rather than your IP address.

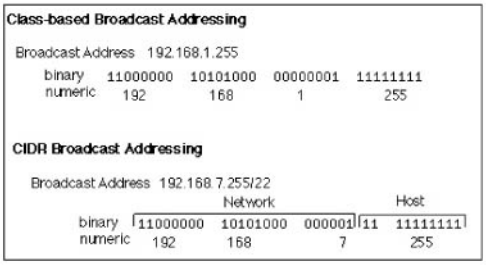


FIGURE 34-3 Class-based and CIDR broadcast addressing

You can also use the domain names of other systems to reference them, so you needn't know their IP addresses. You must know the IP addresses of any domain name servers for your network, however. You can obtain the addresses from your system administrator (often more than one exists). Even if you are using an ISP, you must know the address of the domain name servers your ISP operates for the Internet.

IPv6 Addressing

IPv6 addresses introduce major changes into the format and method of addressing systems under the Internet Protocol (see RFC 3513 at **ietf.org/rfc** or **faqs.org** for more details). There are several different kinds of addressing with different fields for the network segment. The host segment has been expanded to a 64-bit address, allowing direct addressing for a far larger number of systems. Each address begins with a type field specifying the kind of address, which will then determine how its network segment is organized. These changes are designed not only to expand the address space but also to provide greater control over transmissions at the address level.

NOTE *Most distributions already enable IPv6 support in the kernel. Kernel support for IPv6 is provided by the IPv6 kernel module. Kernel configuration support can be found under Device Drivers | Networking Support | Networking Options | The IPv6 Protocol.*

IPv6 Address Format

An IPv6 address consists of 128 bits, up from the 32 bits used in IPv4 addresses. The first 64 bits are used for network addressing, of which the first few bits are reserved for indicating the address type. The last 64 bits are used for the interface address, known as the interface identifier field. The amount of bits used for subnetting can be adjusted with a CIDR mask, much like that in IPv4 CIDR addressing (see the preceding section).

An IPv6 address is written as eight segments representing 16 bits each (128 bits total). To more easily represent 16-bit binary numbers, hexadecimal numbers are used. Hexadecimal numbers use 16 unique numbers, instead of the 8 used in octal numbering. These are 0–9, continuing with the characters A–F.

In the following example, the first four segments represent the network part of the IPv6 address, and the following four segments represent the interface (host) address:

```
FEC0:0000:0000:0000:0008:0800:200C:417A
```

You can cut any preceding zeros, but not trailing zeros, in any given segment. Segments with all zeros can be reduced to a single zero.

```
FEC0:0:0:0:8:800:200C:417A
```

The loopback address used for localhost addressing can be written with seven preceding zeros and a 1.

```
0:0:0:0:0:0:0:1
```

Many addresses will have sequences of zeros. IPv6 supports a shorthand symbol for representing a sequence of several zeros in adjacent fields. This consists of a double colon (::). There can be only one use of the :: symbol per address.

```
FEC0::8:800:200C:417A
```

The loopback address 0000000000000001 can be reduced to just the following:

```
::1
```

To ease the transition from IPv4 addressing to IPv6, a form of addressing incorporating IPv4 addresses is also supported. In this case, the IPv4 address (32 bits) can be used to represent the last two segments of an IPv6 address and can be written using IPv4 notation.

```
FEC0::192.168.0.3
```


IPv6 Interface Identifiers

The identifier part of the IPv6 address takes up the second 64 bits, consisting of four segments containing four hexadecimal numbers. The interface ID is a 64-bit (four-segment) Extended Unique Identifier (EUI-64) generated from a network device's Media Access Control (MAC) address.

IPv6 Address Types

There are three basic kinds of IPv6 addresses, unicast, multicast, and anycast. These, in turn, can have their own types of addresses.

- A *unicast* address is used for a packet that is sent to a single destination.
- An *anycast* address is used for a packet that can be sent to more than one destination.
- A *multicast* address is used to broadcast a packet to a range of destinations.

In IPv6, addressing is controlled by the format prefix that operates as a kind of address type. The format prefix is the first field of the IP address. The three major kinds of unicast network addresses are global, link-local, and site-local, which are each indicated by their own format prefix (see Table 34-5).

- Global addresses begin with the address type 3, site-local with FEC, and link-local with FE8. Global addresses can be sent across the Internet.
- Link-local addresses are used for physically connected systems on a local network.
- Site-local can be used for any hosts on a local network. Site-local addresses operate like IPv4 private addresses; they are used only for local access and cannot be used to transmit over the Internet.

In addition, IPv6 has two special reserved addresses. The address 0000000000000001 is reserved for the loopback address used for a system's localhost address, and the address 0000000000000000 is the unspecified address.

IPv6 Format Prefixes and Reserved Addresses	Description
3	Unicast global addresses
FE8	Unicast link-local addresses, used for physically connected hosts on a network
FEC	Unicast site-local addresses, comparable to IPv4 private addresses
0000000000000001	Unicast loopback address (for system self-communication, localhost)
0000000000000000	Unspecified address
FF	Multicast addresses

Table 34-5 IPv6 Format Prefixes and Reserved Addresses

IPv6 Unicast Global Addresses

IPv6 global addresses currently use four fields: the format prefix, a global routing prefix, the subnet identifier, and the interface identifier. The format prefix for a unicast global address is 3 (3 bits). The global routing prefix references the network address (45 bits), and the subnet ID references a subnet within the site (16 bits).

IPv6 Unicast Local Use Addresses: Link-Local and Site-Local Addresses

For local use, IPv6 provides both link-local and site-local addresses. Link-local addressing is used for interfaces (hosts) that are physically connected to a network. This is usually a small local network. A link-local address uses only three fields, the format prefix FE8 (10 bits), an empty field (54 bits), and the interface identifier (host address, 64 bits). In effect, the network section is empty.

IPv6 site-local addresses have three fields: the format prefix (10 bits), the subnet identifier (54 bits), and the interface identifier (64 bits). Except for any local subnetting, there is no network address.

IPv6 Multicast Addresses

Multicast addresses have a format prefix of FF (8 bits) with flag and scope fields to indicate whether the multicast group is permanent or temporary and whether it is local or global in scope. A group identifier (112 bits) references the multicast group. For the scope, 2 is link-local, 5 is site-local, and E is global. In addition to their interface identifiers, hosts will also have a group ID that can be used as a broadcast address. You use this address to broadcast to the hosts. The following example will broadcast only to those hosts on the local network (5) with the group ID 101:

```
FF05:0:0:0:0:0:0:101
```

To broadcast to all the hosts in a link-local scope, you would use the broadcast address:

```
FF02:0:0:0:0:0:0:1
```

For a site-local scope, a local network, you would use

```
FF05:0:0:0:0:0:0:2
```

IPv6 and IPv4 Coexistence Methods

In the transition from IPv4 to IPv6, many networks will find the need to support both. Some will be connected to networks that use the contrary protocol, and others will have to connect through other network connections that use that protocol. There are several official IETF methods for providing IPv6 and IPv4 cooperation, which fall into three main categories:

- **Dual-stack** Allows IPv4 and IPv6 to coexist on the same networks.
- **Translation** Enables IPv6 devices to communicate with IPv4 devices.
- **Tunneling** Allows transmission from one IPv6 network to another through IPv4 networks, as well as allowing IPv6 hosts to operate on or through IPv4 networks.

In the dual-stack methods, both IPv6 and IPv4 addresses are supported on the network. Applications and DNS servers can use either to transmit data.

Translation uses NAT tables (see Chapter 20) to translate IPv6 addresses to corresponding IPv4 addresses and vice versa as needed. IPv4 applications can then freely interact with IPv6 applications. IPv6-to-IPv6 transmissions are passed directly through, enabling full IPv6 functionality.

Tunneling is used when one IPv6 network needs to transmit to another through an IPv4 network that cannot handle IPv6 addresses. With tunneling, the IPv6 packet is encapsulated within an IPv4 packet, where the IPv4 network then uses the outer IPv4 addressing to pass on the packet. Several methods are used for tunneling, as shown here, as well as direct manual manipulation:

- **6-over-4** Used within a network to use IPv4 multicasting to implement a virtual LAN to support IPv6 hosts, without an IPv6 router (RFC 2529)
- **6-to-4** Used to allow IPv6 networks to connect to and through a larger IPv4 network (the Internet), using the IPv4 network address as an IPv6 network prefix (RFC 3056)
- **Tunnel brokers** Web-based services that create tunnels (RFC 3053)

TCP/IP Configuration Files

A set of configuration files in the **/etc** directory, shown in Table 34-6, is used to set up and manage your TCP/IP network. These configuration files specify such network information as host and domain names, IP addresses, and interface options. The IP addresses and domain names of other Internet hosts you want to access are entered in these files. If you configured your network during installation, you can already find that information in these files.

Identifying Hostnames: /etc/hosts

Without the unique IP address the TCP/IP network uses to identify computers, a particular computer cannot be located. Because IP addresses are difficult to use or remember, domain names are used instead. For each IP address, a domain name exists. When you use a domain

Address	Description
Host address	IP address of your system; it has a network part to identify the network you are on and a host part to identify your own system
Network address	IP address of your network
Broadcast address	IP address for sending messages to all hosts on your network at once
Gateway address	IP address of your gateway system, if you have one (usually the network part of your host IP address with the host part set to 1)
Domain name server addresses	IP addresses of domain name servers your network uses
Netmask	Used to determine the network and host parts of your IP address
File	**Description**
/etc/hosts	Associates hostnames with IP addresses, lists domain names for remote hosts with their IP addresses
/etc/host.conf	Lists resolver options
/etc/nsswitch.conf	Name Switch Service configuration
/etc/resolv.conf	Lists domain name server names, IP addresses (name server), and domain names where remote hosts may be located (search)
/etc/protocols	Lists protocols available on your system
/etc/services	Lists available network services, such as FTP and Telnet, and the ports they use

TABLE 34-6 TCP/IP Configuration Addresses and Files

name to reference a computer on the network, your system translates it into its associated IP address. This address can then be used by your network to locate that computer.

Originally, every computer on the network was responsible for maintaining a list of the hostnames and their IP addresses. This list is still kept in the **/etc/hosts** file. When you use a domain name, your system looks up its IP address in the **hosts** file. The system administrator is responsible for maintaining this list. Because of the explosive growth of the Internet and the development of more and more large networks, the responsibility for associating domain names and IP addresses has been taken over by domain name servers. The **hosts** file is still used to hold the domain names and IP addresses of frequently accessed hosts, however. Your system normally checks your **hosts** file for the IP address of a domain name before taking the added step of accessing a name server.

The format of a domain name entry in the **hosts** file is the IP address followed by the domain name, separated by a space. You can then add aliases for the hostname. After the entry, on the same line, you can enter a comment. A comment is always preceded by a **#** symbol. You can already find an entry in your **hosts** file for localhost.localdomain and

localhost with the IP address 127.0.0.1; localhost is a special identification used by your computer to enable users on your system to communicate locally with each other. The IP address 127.0.0.1 is a special reserved address used by every computer for this purpose. It identifies what is technically referred to as a *loopback device.* The corresponding IPv6 localhost address is **::1** which has the host name **localhost6**. You should never remove the **localhost** and **localhost6** entries. A sample **/etc/hosts** file is shown here:

```
/etc/hosts
127.0.0.1              localhost.localdomain localhost turtle.mytrek.com
::1                    localhost6.localdomain6  localhost6
192.168.0.1            turtle.mytrek.com
192.168.0.2            rabbit.mytrek.com
192.168.34.56          pango1.mytrain.com
```


/etc/resolv.conf

The **/etc/resolv.conf** file holds the IP addresses for your DNS servers along with domains to search. A DNS entry will begin with the term nameserver followed by the name server's IP address. A search entry will list network domain addresses. Check this file to see if your network DNS servers have been correctly listed. If you have a router for a local network, DHCP will automatically place an entry for it in this file. The router in turn will reference your ISP's name server.

```
/etc/resolv.conf
search  mytrek.com   mytrain.com
nameserver  192.168.0.1
nameserver  192.168.0.3
```

NOTE *On Red Hat and Fedora, the* ***/etc/sysconfig/network-scripts*** *directory holds configuration information for different network connection devices such the IP address and network address used.*

/etc/services

The **/etc/services** file lists network services available on your system, such as FTP and Telnet, and associates each with a particular port. Here, you can find out what port your web server is checking or what port is used for your FTP server. You can give a service an alias, which you specify after the port number. You can then reference the service using the alias.

/etc/protocols

The **/etc/protocols** file lists the TCP/IP protocols currently supported by your system. Each entry shows the protocol number, its keyword identifier, and a brief description. See **iana.org/assignments/protocol-numbers** for a complete listing.

Domain Name Service (DNS)

Each computer connected to a TCP/IP network, such as the Internet, is identified by its own IP address. IP addresses are difficult to remember, so a domain name version of each IP address is also used to identify a host. A domain name consists of two parts, the hostname

and the domain. The hostname is the computer's specific name, and the domain identifies the network of which the computer is a part. The domains used for the United States usually have extensions that identify the type of host. For example, **.edu** is used for educational institutions and **.com** is used for businesses. International domains usually have extensions that indicate the country they are located in, such as **.de** for Germany or **.au** for Australia. The combination of a hostname, domain, and extension forms a unique name by which a computer can be referenced. The domain can, in turn, be split into further subdomains.

As you know, a computer on a network can still be identified only by its IP address, even if it has a hostname. You can use a hostname to reference a computer on a network, but this involves using the hostname to look up the corresponding IP address in a database. The network then uses the IP address, not the hostname, to access the computer. Before the advent of large TCP/IP networks, such as the Internet, it was feasible for each computer on a network to maintain a file with a list of all the hostnames and IP addresses of the computers connected on its network. Whenever a hostname was used, it was looked up in this file and the corresponding IP address was located. You can still do this on your own system for remote systems you access frequently.

As networks became larger, it became impractical—and, in the case of the Internet, impossible—for each computer to maintain its own list of all the domain names and IP addresses. To provide the service of translating domain addresses to IP addresses, databases of domain names were developed and placed on their own servers. To find the IP address of a domain name, you send a query to a name server, which then looks up the IP address for you and sends it back. In a large network, several name servers can cover different parts of the network. If a name server cannot find a particular IP address, it sends the query on to another name server that is more likely to have it.

If you are administering a network and you need to set up a name server for it, you can configure a Linux system to operate as a name server. To do so, you must start up a name server daemon and then wait for domain name queries. A name server makes use of several configuration files that enable it to answer requests. The name server software used on Linux systems is the Berkeley Internet Name Domain (BIND) server distributed by the Internet Software Consortium (**isc.org**).

Name servers are queried by resolvers. These are programs specially designed to obtain addresses from name servers. To use domain names on your system, you must configure your own resolver. Your local resolver is configured with your **/etc/host.conf** and **/etc/resolv.conf** files. You can use **/etc/nsswitch** in place of **/etc/host.conf**.

host.conf

Your **host.conf** file lists resolver options (shown in Table 34-7). Each option can have several fields, separated by spaces or tabs. You can use a **#** at the beginning of a line to enter a comment. The options tell the resolver what services to use. The order of the list is important. The resolver begins with the first option listed and moves on to the next ones in turn. You can find the **host.conf** file in your **/etc** directory, along with other configuration files.

In the next example of a **host.conf** file, the **order** option instructs your resolver first to look up names in your local **/etc/hosts** file, and then, if that fails, to query domain name servers. The system does not have multiple addresses.

Option	Description
`order`	Specifies sequence of name resolution methods: `hosts` Checks for name in the local **/etc/host** file `bind` Queries a DNS name server for an address `nis` Uses Network Information Service protocol to obtain an address
`alert`	Checks addresses of remote sites attempting to access your system; you turn it on or off with the `on` and `off` options
`nospoof`	Confirms addresses of remote sites attempting to access your system
`trim`	Checks your localhost's file; removes the domain name and checks only for the hostname; enables you to use only a hostname in your host file for an IP address
`multi`	Checks your localhost's file; allows a host to have several IP addresses; you turn it on or off with the `on` and `off` options

TABLE 34-7 Resolver Options, host.conf

```
/etc/host.conf
# host.conf file
# Lookup names in host file and then check DNS
order bind host
# There are no multiple addresses
multi off
```

/etc/nsswitch.conf: Name Service Switch

Different functions in the standard C Library must be configured to operate on your Linux system. Previously, database-like services, such as password support and name services like NIS or DNS, directly accessed these functions, using a fixed search order. For the GNU C Library 2.*x*, used on current versions of Linux, this configuration is carried out by a scheme called the Name Service Switch (NSS), which is based on the method of the same name used by Sun Microsystems Solaris 2 OS. The database sources and their lookup order are listed in the **/etc/nsswitch.conf** file.

The **/etc/nsswitch.conf** file holds entries for the different configuration files that can be controlled by NSS. The system configuration files that NSS supports are listed in Table 34-8. An entry consists of two fields: the service and the configuration specification. The service consists of the configuration file followed by a colon. The second field is the configuration specification for that file, which holds instructions on how the lookup procedure will work. The configuration specification can contain service specifications and action items. Service specifications are the services to search. Currently, valid service specifications are nis, nisplus, files, db, dns, and compat (see Table 34-9). Not all are valid for each configuration file. For example, the dns service is valid only for the **hosts** file, whereas nis is valid for all files. The following example will first check the local **/etc/password** file and then NIS:

```
passwd:  files nisplus
```

File	Description
aliases	Mail aliases, used by Sendmail
ethers	Ethernet numbers
group	Groups of users
hosts	Hostnames and numbers
netgroup	Networkwide list of hosts and users, used for access rules; C libraries before glibc 2.1 only support netgroups over NIS
network	Network names and numbers
passwd	User passwords
protocols	Network protocols
publickey	Public and secret keys for SecureRPC used by NFS and NIS+
rpc	Remote procedure call names and numbers
services	Network services
shadow	Shadow user passwords

TABLE 34-8 NSS-Supported Files

An action item specifies the action to take for a specific service; it is placed within brackets after a service. A configuration specification can list several services, each with its own action item. In the following example, the entry for the network file has a configuration specification that says to check the NIS and, if not found, to check the **/etc/protocols** file:

```
protocols: nisplus [NOTFOUND=return] files
```

An action item consists of a status and an action. The status holds a possible result of a service lookup, and the action is the action to take if the status is true. Currently, the possible status values are SUCCESS, NOTFOUND, UNAVAIL, and TRYAGAIN

Service	Description
files	Checks corresponding **/etc** file for the configuration (for example, **/etc/hosts** for hosts); this service is valid for all files
db	Checks corresponding **/var/db** databases for the configuration; valid for all files except **netgroup**
compat	Valid only for **passwd**, **group**, and **shadow** files
dns	Checks the DNS service; valid only for **hosts** file
nis	Checks the NIS; valid for all files
nisplus	NIS version 3
hesiod	Uses Hesiod for lookup

TABLE 34-9 NSS Configuration Services

(service temporarily unavailable). The possible actions are return and continue: return stops the lookup process for the configuration file, whereas continue continues on to the next listed service. In the preceding example, if the record is not found in NIS, the lookup process ends.

Shown here is a copy of the **/etc/nsswitch.conf** file, which lists commonly used entries. Comments and commented-out entries begin with a **#** sign:

```
/etc/nsswitch.conf
# /etc/nsswitch.conf
#
# An example Name Service Switch config file.
passwd:                 db files nisplus nis
shadow:                 db files nisplus nis
group:                  db files nisplus nis
hosts:                  files nisplus dns
bootparams:             nisplus [NOTFOUND=return] files
ethers:                 files
netmasks:               files
networks:               files
protocols:              files
rpc:                    files
services:               files
netgroup:               nisplus
publickey:              nisplus
automount:              files
aliases:                files nisplus
```


Network Interfaces and Routes: ifconfig and route

Your connection to a network is made by your system through a particular hardware interface, such as an Ethernet card or a modem. Data passing through this interface is then routed to your network. The **ifconfig** command configures your network interfaces, and the **route** command sets up network connections accordingly. If you configure an interface with a network configuration tool provided by your Linux distribution, you needn't use **ifconfig** or **route**. However, you can directly configure interfaces using **ifconfig** and **route**, if you want. Every time you start your system, the network interfaces and their routes must be established. This is done automatically for you when you boot up by **ifconfig** and **route** commands executed for each interface by the **/etc/rc.d/init.d/network** initialization file, which is executed whenever you start your system. If you are manually adding your own interfaces, you must set up the network script to perform the **ifconfig** and **route** operations for your new interfaces.

ifconfig

The **ifconfig** command takes as its arguments the name of an interface and an IP address, as well as options. The **ifconfig** command then assigns the IP address to the interface. Your system now knows that such an interface exists and that it references a particular IP address. In addition, you can specify whether the IP address is a host address or a network address. You can use a domain name for the IP address, provided the domain name is listed

along with its IP address in the **/etc/hosts** file. The syntax for the **ifconfig** command is as follows:

```
# ifconfig interface -host_net_flag address options
```

The *host_net_flag* can be either **-host** or **-net** to indicate a host or network IP address. The **-host** flag is the default. The **ifconfig** command can have several options, which set different features of the interface, such as the maximum number of bytes it can transfer (**mtu**) or the broadcast address. The **up** and **down** options activate and deactivate the interface. In the next example, the **ifconfig** command configures an Ethernet interface:

```
# ifconfig eth0 192.168.0.1
```

For a simple configuration such as this, **ifconfig** automatically generates a standard broadcast address and netmask. The standard broadcast address is the network address with the number 255 for the host address. For a class C network, the standard netmask is 255.255.255.0, whereas for a class A network, the standard netmask is 255.0.0.0. If you are connected to a network with a particular netmask and broadcast address, however, you must specify them when you use **ifconfig**. The option for specifying the broadcast address is **broadcast**; for the network mask, it is **netmask**. Table 34-10 lists the different **ifconfig** options. In the next example, **ifconfig** includes the netmask and broadcast address:

```
# ifconfig eth0 192.168.0.1 broadcast 192.168.0.255 netmask 255.255.255.0
```

Option	Description
Interface	Name of the network interface, such as **eth0** for the first Ethernet device or **ppp0** for the first PPP device (modem)
up	Activates an interface; implied if IP address is specified
down	Deactivates an interface
allmulti	Turns on or off the promiscuous mode; preceding hyphen (**-**) turns it off; this allows network monitoring
mtu *n*	Maximum number of bytes that can be sent on this interface per transmission
dstaddr *address*	Destination IP address on a point-to-point connection
netmask *address*	IP network mask; preceding hyphen (**-**) turns it off
broadcast *address*	Broadcast address; preceding hyphen (**-**) turns it off
point-to-point *address*	Point-to-point mode for interface; if *address* is included, it is assigned to remote system
hw	Sets hardware address of interface
Address	IP address assigned to interface

TABLE 34-10 The **ifconfig** Options

Once you configure your interface, you can use **ifconfig** with the **up** option to activate it and with the **down** option to deactivate it. If you specify an IP address in an **ifconfig** operation, as in the preceding example, the **up** option is implied.

```
# ifconfig eth0 up
```

Point-to-point interfaces such as Parallel IP (PLIP), Serial Line IP (SLIP), and Point-to-Point Protocol (PPP) require you to include the **pointopoint** option. A PLIP interface name is identified with the name **plip** with an attached number. For example, **plip0** is the first PLIP interface. SLIP interfaces use **slip0**. PPP interfaces start with **ppp0**. Point-to-point interfaces are those that usually operate between only two hosts, such as two computers connected over a modem. When you specify the **pointopoint** option, you need to include the IP address of the host. In the next example, a PLIP interface is configured that connects the computer at IP address 192.168.1.72 with one at 204.166.254.14. If domain addresses are listed for these systems in **/etc/hosts**, those domain names can be used in place of the IP addresses.

```
# ifconfig plip0 192.168.1.72 pointopoint 204.166.254.14
```

If you need to, you can also use **ifconfig** to configure your loopback device. The name of the loopback device is **lo**, and its IP address is the special address 127.0.0.1. The following example shows the configuration:

```
# ifconfig lo 127.0.0.1
```

The **ifconfig** command is useful for checking on the status of an interface. If you enter the **ifconfig** command along with the name of the interface, information about that interface is displayed:

```
# ifconfig eth0
```

To see if your loopback interface is configured, you can use **ifconfig** with the loopback interface name, **lo**:

```
# ifconfig lo
```

Routing

A packet that is part of a transmission takes a certain *route* to reach its destination. On a large network, packets are transmitted from one computer to another until the destination computer is reached. The route determines where the process starts and to what computer your system needs to send the packet for it to reach its destination. On small networks, routing may be static—that is, the route from one system to another is fixed. One system knows how to reach another, moving through fixed paths. On larger networks and on the Internet, however, routing is dynamic. Your system knows the first computer to send its packet off to, and then that computer takes the packet from there, passing it on to another computer, which then determines where to pass it on. For dynamic routing, your system needs to know little. Static routing, however, can become complex because you have to keep track of all the network connections.

Your routes are listed in your routing table in the **/proc/net/route** file. To display the routing table, enter **route** with no arguments (the **netstat -r** command will also display the routing table):

```
# route
Kernel routing table
Destination Gateway      Genmask      Flags Metric Ref Use  Iface
192.168.0.0     *        255.255.255.0  U     0      0   0    etho
127.0.0.0       *        255.0.2055.0   U     0      0   0    lo
default     192.168.0.1  0.0.0.0        UG    0      0   0    eth0
```

Each entry in the routing table has several fields, providing information such as the route destination and the type of interface used. The different fields are listed in Table 34-11.

With the **add** argument, you can add routes either for networks with the **-net** option or with the **-host** option for IP interfaces (hosts). The **-host** option is the default. In addition, you can then specify several parameters for information, such as the netmask (**netmask**), the gateway (**gw**), the interface device (**dev**), and the default route (**default**). If you have more than one IP interface on your system, such as several Ethernet cards, you must specify the name of the interface using the **dev** parameter. If your network has a gateway host, you use the **gw** parameter to specify it. If your system is connected to a network, at least one entry should be in your routing table that specifies the default route. This is the route taken by a message packet when no other route entry leads to its destination. The following example is the routing of an Ethernet interface:

```
# route add 192.168.1.2 dev eth0
```

If your system has only the single Ethernet device as your IP interface, you can leave out the **dev eth0** parameter:

```
# route add 192.168.1.2
```

Field	Description
Destination	Destination IP address of the route
Gateway	IP address or hostname of the gateway the route uses; * indicates no gateway is used
Genmask	The netmask for the route
Flags	Type of route: U = up, H = host, G = gateway, D = dynamic, M = modified
Metric	Metric cost of route
Ref	Number of routes that depend on this one
Window	TCP window for AX.25 networks
Use	Number of times used
Iface	Type of interface this route uses

TABLE 34-11 Routing Table Entries

You can delete any route you establish by invoking **ifconfig** with the **del** argument and the IP address of that route, as in this example:

```
# route del 192.168.1.2
```

For a gateway, you first add a route to the gateway interface, and then add a route specifying that it is a gateway. The address of the gateway interface in this example is 192.168.1.1:

```
# route add 192.168.1.1
# route add default gw 192.168.1.1
```

If you are using the gateway to access a subnet, add the network address for that network (in this example, 192.168.23.0):

```
# route add -net 192.168.23.0 gw dev eth1
```

To add another IP address to a different network interface on your system, use the **ifconfig** and **route** commands with the new IP address. The following command configures a second Ethernet card (**eth1**) with the IP address 192.168.1.3:

```
ifconfig eth1 192.168.1.3
route add 192.168.1.3 dev eth1
```


Wireless Networking

Wireless connections are usually detected and configured automatically by Network Manager (GNOME) or KNetworkManager (KDE). Should you need to, you can also configure a wireless connection manually. A wireless connection operates much like a standard Ethernet connection, requiring only an IP address and DNS server information to connect to the Internet. In addition, wireless connection information is needed such as the network name and channel used. A wireless connection is configured by setting the mode, network name, channel, transmit rate, and key information.

- **Mode** For a simple network, one that does not require roaming, the mode is usually Ad Hoc. Managed networks allow roaming among different access points.
- **Network Name (SSID)** The Network Name is used to identify a cell as part of a virtual network.
- **Channel** Starting from 1, choose one with the least interference.
- **Transmit Rate** Usually set to Auto to adjust automatically to degraded transmissions, but you can set a specific rate such as 11 M or 1 M from the pop-up menu.
- **Key** This is the encryption key for your wireless network. It must be the same for each cell on your network.

Network Manager: GNOME

Most distributions uses Network Manager to detect your network connections, both wired and wireless. Network Manager uses the automatic device detection capabilities of udev and HAL to configure your connections. Once started, Network Manager will display

a Network icon to the right on the top panel. Left-click to see a list of all possible network connections, including all wireless connections available. Right-click to have the option of shutting off your connection (work offline), or to see information about the connection.

NOTE *The KDE version of Network Manager, KNetworkManager, also detects network connections. In addition it allows you to configure PPP dial-up connections as well as manage wireless connections.*

With multiple wireless access points for Internet connections, a system can have several different network connections to choose from, instead of a single-line connection like DSL or cable. This is particularly true for notebook computers that can access different wireless connections at different locations. Instead of manually configuring a new connection each time one is encountered, the Network Manager tool can automatically configure and select a connection to use.

By default, an Ethernet connection will be preferred, if available. Direct lines that support Ethernet connections are normally considered faster than wireless ones. For wireless connections, you will need to choose the one you want.

Network Manager is designed to work in the background, providing status information for your connection and switching from one configured connection to another as needed. For initial configuration, it detects as much information as possible about the new connection. It operates as a GNOME Panel applet, monitoring your connection and can work on any Linux distribution.

Network Manager operates as a daemon with the name NetworkManager. It is managed with the NetworkManager service script.

```
service NetworkManager start
```

If no Ethernet connection is available, Network Manager will scan for a wireless connection and check for Extended Service Set Identifiers (ESSIDs). If an ESSID identifies a previously used connection, then it is automatically selected. If several are found, then the most recently used one is chosen. If only a new connection is available, then Network Manager waits for the user to choose one. A connection is selected only if the user is logged in. If an Ethernet connection is later made, Network Manager will switch to it from wireless.

Network Manager is user specific. When a user logs in, it selects the one preferred by that user. The first time a user runs Network Manager, the notification applet will display a list of current possible connections from which the user can choose.

Clicking the Network Manager icon in the panel will list available network connections. Password-protected access points will display a lock next to them. You will have to configure hidden access points yourself. Select Other Wireless Networks from the applets listing to open a dialog where you can enter the ESSID of the network, the key type, and the password.

Network Interface Connection (NIC cards) hardware is detected using HAL. Information provided by Network Manager is made available to other applications over D-Bus. Features currently under development include VPN and application notification. Network Manager uses the DHCPCD client to gather network information. For user interaction and notification, it uses NetworkManagerInfo.

Manual Wireless Configurations

Network Manager will automatically detect and configure your wireless connections, as will KNetworkManager. However, you can manually configure your connections with wireless tools like `ifwconfig`. Wireless configuration makes use of the same set of wireless extensions. The Wireless Tools package is a set of network configuration and reporting tools for wireless devices installed on a Linux system. They are currently supported and developed as part of the Linux Wireless Extension and Wireless Tools Project, an open source project maintained by Hewlett-Packard.

Wireless Tools consists of the configuration and report tools listed here:

Tool	Description
iwconfig	Sets the wireless configuration options basic to most wireless devices.
iwlist	Displays current status information of a device.
iwspy	Sets the list of IP addresses in a wireless network and checks the quality of their connections.
iwpriv	Accesses configuration options specific to a particular device.

The wireless LAN device will have an Ethernet name just like an Ethernet card. The appropriate modules will automatically be loaded, listing their aliases in the **/etc/modprobe.conf** file.

iwconfig

The **iwconfig** command works much like **ifconfig**, configuring a network connection. It is the tool used by system-config-network to configure a wireless card. Alternatively, you can run **iwconfig** directly on a command line, specifying certain parameters. Added parameters let you set wireless-specific features such as the network name (nwid), the frequency or channel the card uses (freq or channel), and the bit rate for transmissions (rate). See the **iwconfig** Man page for a complete listing of accepted parameters. Some of the commonly used parameters are listed in Table 34-12.

For example, to set the channel used for the wireless device installed as the first Ethernet device, you use the following, setting the channel to 2:

```
iwconfig eth0 channel 2
```

You can also use **iwconfig** to display statistics for your wireless devices, just as **ifconfig** does. Enter the **iwconfig** command with no arguments or with the name of the device. Information such as the name, frequency, sensitivity, and bit rate is listed. Check also **/proc/net/wireless** for statistics. Instead of using **iwconfig** directly to set parameters, you can specify them in the wireless device's configuration file.

iwpriv

The **iwpriv** command works in conjunction with **iwconfig**, allowing you to set options specific to a particular kind of wireless device. With **iwpriv**, you can also turn on roaming or select the port to use. You use the *private-command* parameter to enter the device-specific options. The following example sets roaming on:

```
iwpriv eth0 roam on
```

Parameter	Description
essid	A network name
freq	The frequency of the connection
channel	The channel used
nwid or domain	The network ID or domain
mode	The operating mode used for the device, such as Ad Hoc, Managed, or Auto. Ad Hoc = one cell with no access point, Managed = network with several access points and supports roaming, Master = the node is an access point, Repeater = node forwards packets to other nodes, Secondary = backup master or repeater, Monitor = only receives packets
sens	The sensitivity, the lowest signal level at which data can be received
key or enc	The encryption key used
frag	Cut packets into smaller fragments to increase better transmission
bit or rate	Speed at which bits are transmitted; the auto option automatically falls back to lower rates for noisy channels
ap	A specific access point
power	Power management for wakeup and sleep operations

TABLE 34-12 Commonly Used Parameters

iwspy

Your wireless device can check its connection to another wireless device it is receiving data from, reporting the quality, signal strength, and noise level of the transmissions. Your device can maintain a list of addresses for different devices it may receive data from. You use the **iwspy** tool to set or add the addresses that you want checked. You can list either IP addresses or the hardware versions. A **+** sign will add the address, instead of replacing the entire list:

```
iwspy eth0 +192.168.2.5
```

To display the quality, signal, and noise levels for your connections, you use the **iwspy** command with just the device name:

```
iwspy eth0
```

iwlist

To obtain more detailed information about your wireless device, such as all the frequencies or channels available, you use the **iwlist** tool. Using the device name with a particular parameter, you can obtain specific information about a device, including the frequency, access points, rate, power features, retry limits, and encryption keys used. You can use **iwlist** to obtain information about faulty connections. The following example will list the frequencies used on the **eth0** wireless device.

```
iwlist eth0 freq
```

linux-wlan

The linux-wlan project (**linux-wlan.org**) has developed a separate set of wireless drivers designed for Prism-based wireless cards supporting the new 802.11 wireless standard. The original source code package is available from the linux-wlan site at **linux-wlan.org**. The current package is linux-wlan-ng. You will have to unpack and compile the drivers as noted for source-code software packages in the preceding chapter.

Command Line PPP Access: wvdial

If, for some reason, you have been unable to set up a modem connection on your X Window System, you may have to set it up from the command line interface instead of a desktop. For a dial-up PPP connection, you can use the wvdial dialer, which is an intelligent dialer that not only dials up an ISP service but also performs login operations, supplying your username and password. The wvdial program first loads its configuration from the **/etc/wvdial.conf** file. In here, you can place modem and account information, including the modem speed and serial device, as well as the ISP phone number, username, and password. The **wvdial.conf** file is organized into sections, beginning with a section label enclosed in brackets. A section holds variables for different parameters that are assigned values, such as `username = chris`. The default section holds default values inherited by other sections, so you needn't repeat them. Table 34-13 lists the wvdial variables.

You can use the wvdialconf utility to create a default **wvdial.conf** file for you automatically; wvdialconf will detect your modem and set default values for basic features. You can then edit the **wvdial.conf** file and modify the Phone, Username, and Password entries with your ISP dial-up information. Remove the preceding semicolon (**;**) to unquote the entry. Any line beginning with a semicolon is ignored as a comment.

```
$ wvdialconf
```

You can also create a named dialer, such as *myisp* in the following example. This is helpful if you have different ISPs you log in to. The following example shows the **/etc/wvdial.conf** file:

```
/etc/wvdial.conf [Modem0]
Modem = /dev/ttyS0
Baud = 57600
Init1 = ATZ
SetVolume = 0
Dial Command = ATDT

[Dialer Defaults]
Modem = /dev/ttyS0
Baud = 57600
Init1 = ATZ
SetVolume = 0
Dial Command = ATDT

[Dialer myisp]
Username = chris
Password = mypassword
```

```
Modem = /dev/ttyS0
Phone = 555-5555
Area Code = 555
Baud = 57600
Stupid mode = 0
```

To start wvdial, enter the command **wvdial**, which then reads the connection configuration information from the **/etc/wvdial.conf** file; wvdial dials the ISP and initiates the PPP connection, providing your username and password when requested.

```
$ wvdial
```

You can set up connection configurations for any number of connections in the **/etc/wvdial.conf** file. To select one, enter its label as an argument to the **wvdial** command, as shown here:

```
$ wvdial myisp
```

Variable	Description
Inherits	Explicitly inherits from the specified section. By default, sections inherit from the [Dialer Defaults] section.
Modem	The device wvdial should use as your modem. The default is **/dev/modem**.
Baud	The speed at which wvdial communicates with your modem. The default is 57,600 baud.
Init1...Init9	Specifies the initialization strings to be used by your modem; wvdial can use up to 9. The default is ATZ for Init1.
Phone	The phone number you want wvdial to dial.
Area Code	The area code, if any.
Dial Prefix	Any needed dialing prefix—for example, 70 to disable call waiting or 9 for an outside line.
Dial Command	The dial operation. The default is ATDT.
Login	The username you use at your ISP.
Login Prompt	If your ISP has an unusual login prompt, you can specify it here.
Password	The password you use at your ISP.
Password Prompt	If your ISP has an unusual password prompt, you can specify it here.
Force Address	A static IP address to use (for ISPs that provide static IP addresses to users).
Stupid Mode	In Stupid Mode, wvdial does not attempt to interpret any prompts from the terminal server and starts pppd after the modem connects.
Auto Reconnect	If enabled, wvdial attempts to reestablish a connection automatically if you are randomly disconnected by the other side. This option is on by default.

TABLE 34-13 Variables for **wvdial**

Monitoring Your Network: ping, netstat, tcpdump, EtherApe, Ettercap, and Wireshark

Several applications are available on Linux to let you monitor your network activity. Graphical applications like EtherApe, Ettercap, and Wireshark provide detailed displays and logs to let you analyze and detect network usage patterns. Other tools like ping offer specific services.

The EtherApe, Ettercap, and Wireshark tools can be accessed on the Applications | Internet menu. Tools like ping, traceroute, and netstat can be accessed with the GNOME Network Tools application accessible on the Applications | System Tools menu, as well as being rung individually on a command line (terminal window).

EtherApe provides a simple graphical display for your protocol activity. The Preferences dialog lets you set features like the protocol to check and the kind of traffic to report.

ping

With the ping program, you can check to see if you can access another host on your network. The ping program sends a request to the host for a reply. The host then sends a reply back, and it is displayed on your screen. The ping program continually sends such a request until you stop it with a **break** command, CTRL-C. You see one reply after another scroll by on your screen until you stop the program. If ping cannot access a host, it issues a message saying the host is unreachable. If ping fails, this may be an indication that your network connection is not working. It may be only the particular interface, a basic configuration problem, or a bad physical connection. The ping utility uses the Internet Control Message Protocol (ICMP), discussed in Chapter 20. Networks may block these protocols as a security measure, also preventing ping from working. A ping failure may simply indicate a security precaution on the part of the queried network.

To use ping, enter **ping** and the name of the host.

```
$ ping ftp.redhat.com
```

Ettercap

Ettercap is a sniffer program designed to detect Man-in-the-Middle attacks. In this kind of attack, packets are detected and modified in transit to let an unauthorized user access a network. You can use either its graphical interface or its command line interface. Ettercap can perform Unified sniffing on all connections, or Bridged sniffing on a connection between network interfaces. Ettercap uses plug-ins for specific tasks, like dos_attack to detect denial-of-service attacks and dns-spoof for DNS spoofing detection. Check the plug-ins Help panel, or enter **ettercap -P list** for a complete listing. Ettercap can be run in several modes, including a text mode, a command line cursor mode, a script mode using commands in a file, and even as a daemon logging results automatically.

Wireshark

Wireshark is a network protocol analyzer that lets you capture packets transmitted across your network, selecting and examining those from protocols you want to check. You can examine packets from particular transmissions, displaying the data in readable formats. The Wireshark interface displays three panes: a listing of current packets, the protocol tree for

the currently selected packet, and a display of the selected packet contents. The first pane categorizes entries by time, source, destination, and protocol. There are button headers for each. To sort a set of entries by a particular category, click its header. For example to, group entries by protocol, click the Protocol button; for destinations, use the Destination button.

Capture Options

To configure Wireshark, you select the Options entry from the Capture menu. This opens an Options window where you can select the network interface to watch. Here you can also select options such as the file to hold your captured information in and a size limit for the capture, along with a filter to screen packets. With the Promiscuous mode selected, you can see all network traffic passing through that device, whereas with it off, you will see only those packets destined for that device. You can then click the Start button to start Wireshark. To stop and start Wireshark, you select the Stop and Start entries on the Capture menu.

- The Capture Files option lets you select a file to save your capture in. If no file is selected, data is simply displayed in the Wireshark window. If you want to keep a continuous running snapshot of your network traffic, you can use ring buffers. These are a series of files that are used to save captured data. When they fill up, the capture begins saving again to the first file, and so on. Check "Use multiple files" to enable this option.
- Display options control whether packets are displayed in real time on the Wireshark window.
- Limits let you set a limit for the capture packet size.
- Capture filter lets you choose the type of protocol you want to check.
- Name resolution enables the display of host and domain names instead of IP addresses, if possible.

Wireshark Filters

A filter lets you select packets that match specified criteria, such as packets from a particular host. Criteria are specified using expressions supported by the Packet Capture Library and implemented by **`tcpdump`**. Wireshark filters use similar expressions as those used by the **`tcpdump`** command. Check the **`tcpdump`** Man page for detailed descriptions.

You can set up a either a Search filter in the Find panel (Edit menu) to search for certain packets, or set up a Capture filter in the Options panel (Capture menu) to select which packets to record. The filter window is the same for both. On the filter window you can select the protocol you want to search or capture. The filter name and string will appear in the Properties segment. You can also enter your own string, setting up a new filter of your own. The string must be a filter expression.

To create a new filter, enter the name you want to give it in the Filter Name box. Then in the Filter String box, enter the filter expression, like **icmp**, and click New. Your new filter will appear in the list. To change a filter, select it and change its expression in the Filter String box, then click Change.

A filter expression consists of an ID, such as the name or number of the host, and a qualifier. Qualifiers come in three types: type, direction, and protocol. The type can reference

the host, network, or port. The type qualifiers are **host**, **net**, and **port**. Direction selects either source or destination packets, or both. The source qualifier is **src**, and the destination **dst**. With no destination qualifier, both directions are selected. Protocol lets you specify packets for a certain protocol. Protocols are represented using their lowercase names, such as **icmp** for ICMP. For example, the expression to list all packets coming in from a particular host would be **src host** *hostname,* where *hostname* is the source host. The following example will display all packets from the 192.168.0.3 host:

```
src host 192.168.0.3
```

Using just **host** will check for all packets going out as well as coming in for that host. The **port** qualifier will check for packets passing through a particular port. To check for a particular protocol, you use the protocol name. For example, to check for all ICMP packets you would use the expression

```
icmp
```

There are also several special qualifiers that let you further control your selection. The **gateway** qualifier lets you detect packets passing through a gateway. The **broadcast** and **multi-cast** qualifiers detect packets broadcast to a network. The **greater** and **less** qualifiers can be applied to numbers such as ports or IP addresses.

You can combine expressions into a single complex Boolean expression using **and**, **or**, or **not**. This lets you create a more refined filter. For example, to capture only the ICMP packets coming in from host 192.168.0.2, you can use

```
src host 192.168.0.3 and icmp
```

tcpdump

Like Wireshark, **tcpdump** will capture network packets, saving them in a file where you can examine them. **tcpdump** operates entirely from the command line. You will have to open a terminal window to run it. Using various options, you can refine your capture, specifying the kinds of packets you want. **tcpdump** uses a set of options to specify actions you want to take, which include limiting the size of the capture, deciding which file to save it to, and choosing any filter you want to apply to it. Check the **tcpdump** Man page for a complete listing.

- The **-i** option lets you specify an interface to listen to.
- With the **-c** option, you can limit the number of packets to capture.
- Packets will be output to the standard output by default. To save them to a file, you can use the **-w** option.
- You can later read a packet file using the **-r** option and apply a filter expression to it.

The **tcpdump** command takes as its argument a filter expression that you can use to refine your capture. Wireshark uses the same filter expressions as **tcpdump** (see the filters discussion in Wireshark).

netstat

The netstat program provides real-time information on the status of your network connections, as well as network statistics and the routing table. The **netstat** command has several options you can use to bring up different sorts of information about your network:

```
# netstat
Active Internet connections
Proto Recv-Q Send-Q Local Address Foreign Address (State) User
tcp 0 0 turtle.mytrek.com:01 pango1.mytrain.com.:ftp ESTABLISHED dylan
Active UNIX domain sockets
Proto RefCnt Flags Type State Path
unix 1 [ ACC ] SOCK_STREAM LISTENING /dev/printer
unix 2 [ ] SOCK_STREAM CONNECTED /dev/log
unix 1 [ ACC ] SOCK_STREAM LISTENING /dev/nwapi
unix 2 [ ] SOCK_STREAM CONNECTED /dev/log
unix 2 [ ] SOCK_STREAM CONNECTED
unix 1 [ ACC ] SOCK_STREAM LISTENING /dev/log
```

The **netstat** command with no options lists the network connections on your system. First, active TCP connections are listed, and then the active domain sockets are listed. The domain sockets contain processes used to set up communications among your system and other systems. You can use **netstat** with the **-r** option to display the routing table, and **netstat** with the **-i** option displays the uses of the different network interfaces.

IP Aliasing

In some cases, you may want to assign a single Linux system that has only one network interface to two or more IP addresses. For example, you may want to run different websites that can be accessed with separate IP addresses on this same system. In effect, you are setting up an alias for your system, another address by which it can be accessed. In fact, you are assigning two IP addresses to the same network interface—for example, assigning a single Ethernet card two IP addresses. This procedure, referred to as *IP aliasing*, is used to set up multiple IP-based virtual hosts for Internet servers. This method enables you to run several web servers on the same machine using a single interface (or more than one on each of several interfaces). See Chapters 22 and 23 for FTP and web server information about virtual hosts.

Setting up an IP alias is a simple matter of configuring a network interface on your system to listen for the added IP address. Your system needs to know what IP addresses it should listen for and on what network interface. You set up IP aliases using the **ifconfig** and **route** commands or a network administrative tool.

To add another address to the same interface, you need to qualify the interface by adding a colon and a number. For example, to add another IP address to the first Ethernet card (**eth0**), you add a **:0** to its interface name, **eth0:0**. The following example shows the **ifconfig** and **route** commands for the Ethernet interface 192.168.1.2 and two IP aliases added to it: 192.168.1.100 and 192.168.1.101. To add yet another IP address to this same interface, you use **eth0:1**, incrementing the qualifier, and so on. The first **ifconfig** command assigns the main IP address, 192.168.1.2, to the first Ethernet device, **eth0**. Then, two other IP addresses are assigned to that same device. In the first **route** command, the

network route is set up for the Ethernet device, and then routes are set up for each IP interface. The interfaces for the two aliases are indicated with **eth0:0** and **eth0:1**:

```
ifconfig eth0 192.168.1.2
ifconfig eth0:0 192.168.1.100
ifconfig eth0:1 192.168.1.101
route add -net 192.168.1.0 dev eth0
route add -host 192.168.1.2 dev eth0
route add -host 192.168.1.100 dev eth0:0
route add -host 192.168.1.101 dev eth0:1
```

IP aliasing must be supported by the kernel before you can use it. If your kernel does not support it, you may have to rebuild the kernel (including IP aliasing support), or use loadable modules to add IP aliasing.

InfiniBand Support

Linux, since the 2.6.10 kernel, includes InfiniBand support. This is a new I/O architecture that is used to replace the older bus architectures used in current systems. Often, InfiniBand is used as a replacement for local network connections. It is currently implemented in supercomputer and network server clusters. You can find more about InfiniBand at the Linux InfiniBand Project at **infiniband.sourceforge.net**. Support for InfiniBand is being carried out as an open source project by the OpenFabrics Alliance, **openib.org**.

Systems today use the PCI bus or its enhanced versions, PCI X or PCI Express. This PCI I/O architecture uses a shared bus that can only reach about a half gigabit of throughput. Clustered servers are already reaching the limits of this I/O method. One alternative technology is the InfiniBand I/O architecture. InfiniBand uses serial channels instead of a shared bus. Speeds start at 2.6 Gb per second and go as high as 30 Gb per second. Instead of having a bus processing transactions controlled by a single host, InfiniBand uses peer-to-peer channel architecture where multiple connections can be managed using different channels. This fabric switch architecture enables InfiniBand to switch among different nodes. PCI Express is limited to use as a local bus, connecting a CPU with peripherals. InfiniBand, by contrast, supports networking connections, letting you implement essentially a local high-speed intranet as well as shared high-speed connections to standalone storage devices like hard drives. Using an InfiniBand cable instead of an Ethernet cable, you can connect your hosts and shared devices (within up to 50 feet). The IPoIB (IP over InfiniBand) protocol lets you implement IP networking over InfiniBand connections, and the RDMA protocol can be used for remote storage devices. The higher speed of an InfiniBand connection is particularly important for servers needing high-bandwidth capability. In addition, the Sockets Direct Protocol can set up high-speed InfiniBand connections for streams, and the SCSI RDMA Protocol (SRP) manages connections to hard drives.

Machines with PCI Express can handle the greater bandwidth provided by an InfiniBand connection. A Host Channel Adapter (HCA) card placed in a PCI slot has InfiniBand connectors and will interface InfiniBand transmissions with the PCI Express bus. Drivers for several HCAs are already incorporated in the kernel, as are protocol drivers.

This page intentionally left blank

CHAPTER 35

Network Autoconfiguration with IPv6, DHCPv6, and DHCP

Many networks now provide either IPv6 autoconfiguration or the DHCP (Dynamic Host Configuration Protocol) service, which automatically provides network configuration for all connected hosts. Autoconfiguration can be either stateless, as in the case of IPv6, or stateful, as with DHCP. Stateless IPv6 autoconfiguration requires no independent server or source to connect to a network. It is a direct plug-and-play operation, where the hardware network interfaces and routers can directly determine the correct addresses. DCHP is an older method that requires a separate server to manage and assign all addresses. Should this server ever fail, hosts cannot connect.

With the DHCP protocol, an administrator uses a pool of IP addresses from which the administrator can assign an IP address to a host as needed. The protocol can also be used to provide all necessary network connection information such as the gateway address for the network or the netmask. Instead of having to configure each host separately, network configuration can be handled by a central DHCP server. The length of time that an address can be used can be controlled by means of leases, making effective use of available addresses. If your network is configuring your systems with DHCP, you will not have to configure it.

There are currently two versions of DHCP, one for the original IPv4 protocol and another, known as DHCPv6, for the IPv6 protocol. The IPv6 protocol includes information for dynamic configuration that the IPv4 protocol lacks. In this respect, the IPv4 protocol is much more dependent on DHCP than IPv6 is.

IPv6 Stateless Autoconfiguration

In an IPv6 network, the IPv6 protocol includes information that can directly configure a host. With IPv4 you either had to manually configure each host or rely on a DHCP server to provide configuration information. With IPv6, configuration information is integrated into the Internet protocol directly. IPv6 address autoconfiguration is described in detail in RFC 2462.

IPv6 autoconfiguration capabilities are known as stateless, meaning that it can directly configure a host without recourse to an external server. Alternatively, DHCP, including DHCPv6, is stateful, where the host relies on an external DHCP server to provide configuration information. Stateless autoconfiguration has the advantage of hosts not

having to rely on a DHCP server to maintain connections to a network. Networks can even become mobile, hooking into one subnet or another, automatically generating addresses as needed. Hosts are no longer tied to a particular DHCP server.

Generating the Local Address

To autoconfigure hosts on a local network, IPv6 makes use of the each network device's hardware MAC address. This address is used to generate a temporary address, with which the host can be queried and configured.

The MAC address is used to create a link-local address, one with a link-local prefix, **FE80::0**, followed by an interface identifier. The link-local prefix is used for physically connected hosts such as those on a small local network.

A uniqueness test is then performed on the generated address. Using the Neighbor Discovery Protocol (NDP), other hosts on the network are checked to see if another host is already using the generated link-local address. If no other host is using the address, then the address is assigned for that local network. At this point the host has only a local address valid within the local physical network. Link-local addresses cannot be routed to a larger network.

Generating the Full Address: Router Advertisements

Once the link-local address has been determined, the router for the network is queried for additional configuration information. The information can be either stateful or stateless, or both. For stateless configuration, information such as the network address is provided directly, whereas for stateful configuration, the host is referred to a DHCPv6 server where it can obtain configuration information. The two can work together. Often the stateless method is used for addresses, and the stateful DHCPv6 server is used to provide other configuration information such as DNS server addresses.

In the case of stateless addresses, the router provides the larger network address, such as the network's Internet address. This address is then added to the local address, replacing the original link-local prefix, giving either a complete global Internet address or, in the case of private networks, site-local addresses. Routers will routinely advertise this address information, though it can also be specifically requested. The NDP is used to query the information. Before the address is officially assigned, a duplicate address detection procedure checks to see if the address is already in use. The process depends on the router's providing the appropriate addressing information in the form of router advertisements. If there is no router, or there are no route advertisements, then a stateful method like DHCPv6 or manual configuration must be used to provide the addresses.

Figure 35-1 shows a network that is configured with stateless address autoconfiguration. Each host first determines its interface identifier using its own MAC hardware address to create a temporary link-local address for each host using the **FE08::0** prefix. This allows initial communication with the network's router. The router then uses its network prefix to create full Internet addresses, replacing the link-local prefix.

Router Renumbering

With IPv6, routers have the ability to renumber the addresses on their networks by changing the network prefix. Renumbering is carried out through the Router Renumbering (RR) Protocol. (See RFC 2894 for a description of router renumbering.) Renumbering is often

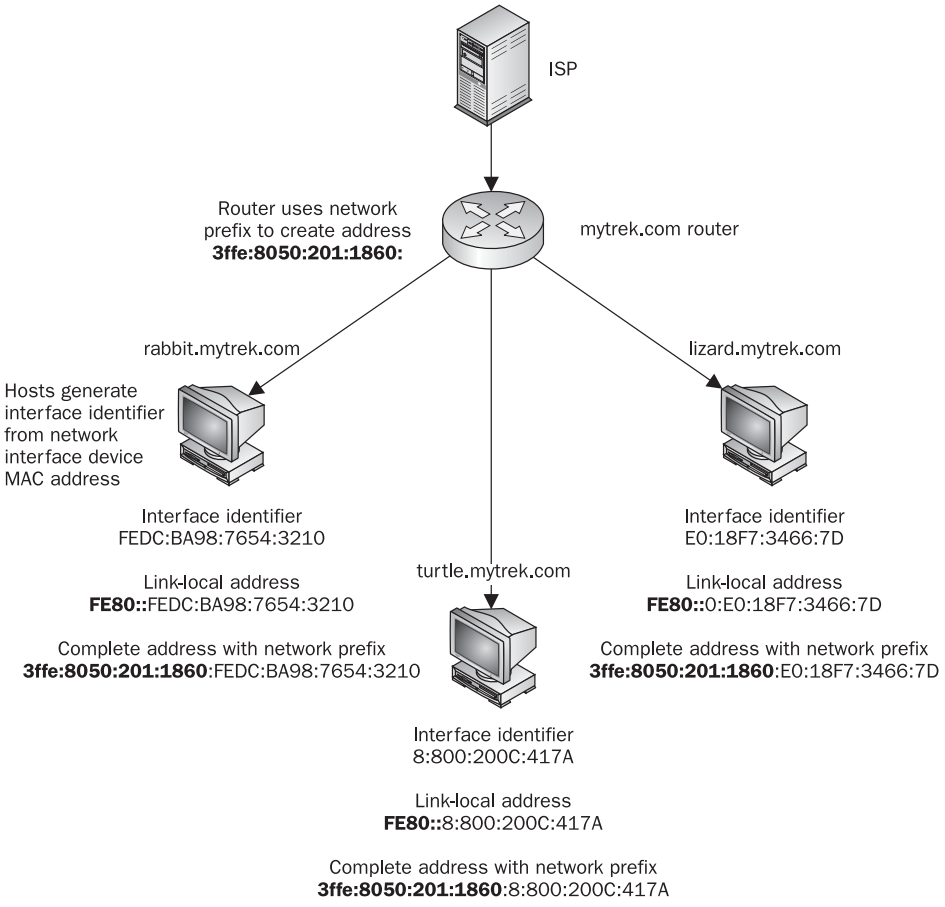


Figure 35-1 Stateless IPv6 address autoconfiguration

used when a network changes ISP providers and requires that the net address for all hosts be changed (see Figure 35-2). It can also be used for mobile networks where a network can be plugged in to different larger networks, renumbering each time.

With renumbering, routers place a time limit on addresses, similar to the lease time in DHCP, by specifying an expiration limit for the network prefix when the address is generated. To ease transition, interfaces still keep their old addresses as deprecated addresses, while the new one are first being used. The new ones will be the preferred addresses used for any new connections, while deprecated ones are used for older connections. In effect, a host can have two addresses, one deprecated and one preferred. This regeneration of addresses effectively renumbers the hosts.

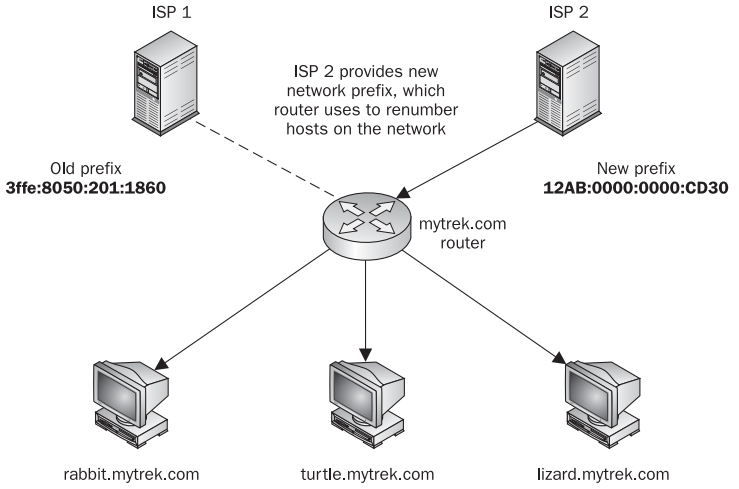


FIGURE 35-2 Router renumbering with IPv6 autoconfiguration

IPv6 Stateful Autoconfiguration: DHCPv6

The IPv6 version of DHCP (DHCPv6) provides stateful autoconfiguration to those networks that still want a DHCP-like service on IPv6 networks. DHCP IPv6 provides configuration information from a server, just like DHCP, but it is a completely different protocol from the IPv4 version, with different options and capabilities. As a stateful configuration process, information is provided by an independent server.

DHCPv6 uses a its own set of options for both the client and the server. The DHCP server for IPv6 is called **dhcp6s**, and the DHCP client for IPv6 is **dhcp6c**. Their corresponding configuration files are **/etc/dhcp6c** and **/etc/sysconfig/dhcp6s**. A service script, **/etc/init.d/dhcp6s**, can be used to manage the **dhcp6s** server.

As with IPv6 autoconfiguration, the host identifier for a local address is first automatically generated. This is a link-local address containing a host identifier address generated from the host interface's MAC address.

Once the link-local address is determined, the router is queried for the DHCPv6 server. This information is provided in router advertisements that are broadcast regularly. At this point the two different kinds of stateful information can be provided by the server: addresses and other configuration information. The host is notified which kinds of stateful information are provided. If address information is not given by the DHCPv6 server, then addresses will be determined using the stateless autoconfiguration method described in the preceding section. If address information is provided, then an address will be obtained from the server instead of being directly generated. Before leasing an address, the server will run a duplicate address detection procedure to make sure the address is unique.

NOTE *DHCPv6 stateful addressing is useful for situations in which strict control needs to be maintained over the IP address of a host, whereas stateless IPv6 addressing is suitable for situations where the actual IP address is not important, just that connections be effective.*

Linux as an IPv6 Router: radvd

For a Linux system that operates as a router, you use the **radvd** (Router ADVertisement Daemon) to advertise addresses, specifying a network prefix in the **/etc/radvd.conf** file. The **radvd** daemon will detect router network address requests from hosts, known as router solicitations, and provide them with a network address using a router advertisement. These router advertisements will also be broadcast to provide the network address to any hosts that do not send in requests. For **radvd** to work, you will have to turn on IPv6 forwarding. Use **sysctl** and set **net.ipv6.conf.all.forwarding** to 1. To start up the **radvd** daemon, you use the **radvd** startup script. To check the router addresses **radvd** is sending, you can use **radvdump**.

You will have to configure the **radvd** daemon yourself, specifying the network address to broadcast. Configuration, though, is very simple, as the full address will be automatically generated using the host's hardware address. A configuration consists of interface entries, which in turn lists interface options, prefix definitions, and options, along with router definitions if needed. The configuration is placed in the **/etc/radvd.conf** file, which will look something like this:

```
interface eth0 {
     AdvSendAdvert on;
          prefix fec0:0:0:0::/64
          {
          AdvOnLink on;
          AdvAutonomous on;
          };
};
```

This assumes one interface is used for the local network, **eth0**. This interface configuration lists an interface option (AdvSendAdvert) and a prefix definition, along with two prefix options (AdvOnLink and AdvAutonomous). To specify prefix options for a specific prefix, add them within parentheses following the prefix definition. The prefix definition specifies your IPv6 network address. If a local area network has its own network address, you will need to provide its IPv6 network prefix address. For a private network, such as a home network, you can use the site-local IPv6 prefix, which operates like the IPv4 private network addresses, 192.168.0. The preceding example uses a site-local address that is used for private IPv6 networks, fec0:0:0:0::, which has a length of 64 bits.

The AdvSendAdvert interface option turns on network address advertising to the hosts. The AdvAutonomous network prefix option provides automatic address configuration, and AdvOnLink simply means that host requests can be received on the specified network interface.

A second network interface is then used to connect the Linux system to an ISP or larger network. If the ISP supports IPv6, this is simply a matter of sending a router solicitation to the ISP router. This automatically generates your Internet address using the hardware address of the network interface that connects to the Internet and the ISP router's advertised network address. In Figure 35-2, shown earlier, the **eth0** network interface connects to the local network, whereas **eth1** connects to the Internet.

DHCP for IPv4

DHCP provides configuration information to systems connected to a IPv4 TCP/IP network, whether the Internet or an intranet. The machines on the network operate as DHCP clients, obtaining their network configuration information from a DHCP server on their network. A machine on the network runs a DHCP client daemon that automatically receives its network configuration information from its network's DHCP server. The information includes its IP address, along with the network's name server, gateway, and proxy addresses, including the netmask. Nothing has to be configured manually on the local system, except to specify the DHCP server it should get its network configuration from. This has the added advantage of centralizing control over network configuration for the different systems on the network. A network administrator can manage the network configurations for all the systems on the network from the DHCP server.

NOTE *DHCP is based on the BOOTP developed for diskless workstations. Check DHCP documentation for options specific to machines designed to work with BOOTP.*

A DHCP server also supports several methods for IP address allocation: automatic, dynamic, and manual. Automatic allocation assigns a permanent IP address for a host. Manual allocation assigns an IP address designated by the network administrator. With dynamic allocation, a DHCP server can allocate an IP address to a host on the network only when the host actually needs to use it. Dynamic allocation takes addresses from a pool of IP addresses that hosts can use when needed and release when they are finished.

The current version of DHCP now supports the DHCP failover protocol, in which two DHCP servers support the same address pool. Should one fail, the other can continue to provide DHCP services for a network. Both servers are in sync and have the same copy of network support information for each host on the network. Primary and secondary servers in this scheme are designated with the primary and secondary statements.

A variety of DHCP servers and clients are available for different operating systems. For Linux, you can obtain DHCP software from the Internet Software Consortium (ISC) at **isc.org**. The software package includes a DHCP server, a client, and a relay agent. Linux includes a DHCP server and client. The DHCP client is called **dhclient**, and the IPv4 server is called **dhcpd**.

Configuring DHCP IPv4 Client Hosts

Configuring hosts to use a DHCP server is a simple matter of setting options for the host's network interface device, such as an Ethernet card. For a Linux host, you can use a distribution network tool to set the host to automatically access a DHCP server for network information. On a network tool's panel for configuring the Internet connection, you will normally find a check box for selecting DHCP. Clicking this box will enable DHCP.

Client support is carried out by the **dhclient** tool. When your network starts up, it uses **dhclient** to set up your DHCP connection. Though defaults are usually adequate, you can further configure the DHCP client using the **/etc/dhclient.conf** file. Consult the **dhclient.conf** Man page for a detailed list of configuration options. **dhclient** keeps lease information on the DCHP connection in the **/var/lib/dhcp/dhclient.leases** file. You can also directly run **dhclient** to configure DHCP connections.

```
dhclient
```

Configuring the DHCP IPv4 Server

You can stop and start the DHCP server using the **dhcpd** command in the **/etc/rc.d/init.d** directory. Use the redhat-config-services tool or the **service** command with the **start**, **restart**, and **stop** options. The following example starts the DHCP server. Use the **stop** option to shut it down and **restart** to restart it.

```
service dhcpd start
```

Dynamically allocated IP addresses, known as *leases*, will be assigned for a given time. When a lease expires, it can be extended or a new one generated. Current leases are listed in the **dhcpd.leases** file located in the **/var/lib/dhcp** directory. A lease entry will specify the IP address and the start and end times of the lease along with the client's hostname.

DHCP server arguments and options can be specified in the **/etc/sysconfig/dhcpd** file. Network device arguments specify which network device the DHCP server should run on. You can also specify options such as the configuration file to use or the port to listen on. Network device arguments are needed should you have two or more network interfaces on your system but want the DHCP server to operate on only selected connections. Such arguments are listed in the **/etc/sysconf/dhcpd** file using the DHCPARGS setting. The following example says to run the DHCP server only on the Ethernet network device **eth0**:

```
DHPCARGS=eth0
```

This kind of configuration is useful for gateway systems that are connected to both a local network and a larger network, such as the Internet, through different network devices. On the Internet connection, you may want to run the DCHP client to receive an IP address from an ISP, and on the local network connection, you would want to run the DHCP server to assign IP addresses to local hosts.

The configuration file for the DHCP server is **/etc/dhcpd.conf**, where you specify parameters and declarations that define how different DHCP clients on your network are accessed by the DHCP server, along with options that define information passed to the clients by the DHCP server. These parameters, declarations, and options can be defined globally for certain subnetworks or for specific hosts. Global parameters, declarations, and options apply to all clients, unless overridden by corresponding declarations and options in subnet or host declarations. Technically, all entries in a **dhcpd.conf** file are statements that can be either declarations or parameters. All statements end with a semicolon. Options are specified in **options** parameter statements. Parameters differ from declarations in that they define if and how to perform tasks, such as how long a lease is allocated. Declarations describe network features such as the range of addresses to allocate or the networks that are accessible. See Table 35-1 for a listing of commonly used declarations and options.

Declarations provide information for the DHCP server or designate actions it is to perform. For example, the **range** declaration is used to specify the range of IP addresses to be dynamically allocated to hosts:

```
range 192.168.0.5 192.168.0.128;
```

With parameters, you can specify how the server is to treat clients. For example, the **default-lease-time** declaration sets the number of seconds a lease is assigned to a client. The **filename** declaration specifies the boot file to be used by the client. The **server-name**

Declarations	Description
shared-network *name*	Indicates if some subnets share the same physical network.
subnet *subnet-number netmask*	References an entire subnet of addresses.
range [*dynamic-bootp*] *low-address* [*high-address*] ;	Provides the highest and lowest dynamically allocated IP addresses.
host *hostname*	References a particular host.
group	Lets you label a group of parameters and declarations and then use the label to apply them to subnets and hosts.
allow unknown-clients; **deny unknown-clients;**	Does not dynamically assign addresses to unknown clients.
allow bootp; deny bootp;	Determines whether to respond to **bootp** queries.
allow booting; deny booting;	Determines whether to respond to client queries.
Parameters	
default-lease-time *time*;	Assigns length in seconds to a lease.
max-lease-time *time*;	Assigns maximum length of lease.
hardware *hardware-type hardware-address*;	Specifies network hardware type (Ethernet or token ring) and address.
filename "*filename*";	Specifies name of the initial boot file.
server-name "*name*";	Specifies name of the server from which a client is booting.
next-server *server-name*;	Specifies server that loads the initial boot file specified in the filename.
fixed-address *address* [, *address ...*] ;	Assigns a fixed address to a client.
get-lease-hostnames *flag*;	Determines whether to look up and use IP addresses of clients.
authoritative; **not authoritative;**	Denies invalid address requests.
server-identifier hostname;	Specifies the server.
Options	
option subnet-mask *ip-address*;	Specifies client's subnet mask.
option routers *ip-address* [, *ip-address...*] ;	Specifies list of router IP addresses on client's subnet.
option domain-name-servers *ip-address* [, *ip-address...*] ;	Specifies list of domain name servers used by the client.

Table 35-1 DHCP Declarations, Parameters, and Options

Options	Description
option log-servers *ip-address* [, *ip-address...*];	Specifies list of log servers used by the client.
option host-name *string*;	Specifies client's hostname.
option domain-name *string*;	Specifies client's domain name.
option broadcast-address *ip-address*;	Specifies client's broadcast address.
option nis-domain *string*;	Specifies client's Network Information Service domain.
option nis-servers *ip-address* [, *ip-address...*];	Specifies NIS servers the client can use.
option smtp-server *ip-address* [, *ip-address...*];	Lists SMTP servers used by the client.
option pop-server *ip-address* [, *ip-address...*];	Lists POP servers used by the client.
option nntp-server *ip-address* [, *ip-address...*];	Lists NNTP servers used by the client.
option www-server *ip-address* [, *ip-address...*];	Lists web servers used by the client.

TABLE 35-1 DHCP Declarations, Parameters, and Options (*continued*)

declaration informs the client of the host from which it is booting. The **fixed-address** declaration can be used to assign a static IP address to a client. See the Man page for **dhcpd.conf** for a complete listing.

Options provide information to clients that they may need to access network services, such as the domain name of the network, the domain name servers that clients use, or the broadcast address. See the Man page for **dhcp-options** for a complete listing. This information is provided by **option** parameters as shown here:

```
option broadcast-address 192.168.0.255;
option domain-name-servers 192.168.0.1, 192.168.0.4;
option domain-name "mytrek.com";
```

Your **dhcpd.conf** file will usually begin with declarations, parameters, and options that you define for your network serviced by the DHCP server. The following example provides router (gateway), netmask, domain name, and DNS server information to clients. Additional parameters define the default and maximum lease times for dynamically allocated IP addresses.

```
option routers 192.168.0.1;
option subnet-mask 255.255.255.0;
option domain-name "mytrek.com";
option domain-name-servers 192.168.0.1;
default-lease-time 21600;
max-lease-time 43200;
```

With the subnet, host, and group declarations, you can reference clients in a specific network, particular clients, or different groupings of clients across networks. Within these declarations, you can enter parameters, declarations, or options that will apply only to those clients. Scoped declarations, parameters, and options are enclosed in braces. For example, to define a declaration for a particular host, you use the **host** declaration as shown here:

```
host rabbit {
        declarations, parameters, or options;
        }
```

You can collect different subnet, global, and host declarations into groups using the **group** declaration. In this case, the global declarations are applied only to those subnets and hosts declared within the group.

Dynamic IPv4 Addresses for DHCP

Your DHCP server can be configured to select IP addresses from a given range and assign them to different clients. Given a situation where you have many clients that may not always be connected to the network, you can effectively service them with a smaller pool of IP addresses. IP addresses are assigned only when they are needed. With the **range** declaration, you specify a range of addresses that can be dynamically allocated to clients. The declaration takes two arguments, the first and last addresses in the range.

```
range 192.168.1.5 192.168.1.128;
```

For example, if you are setting up your own small home network, you would use a network address beginning with 192.168. The range would specify possible IP addresses within that network. So, for a network with the address 192.168.0.0, you place a **range** declaration along with any other information you want to give to your client hosts. In the following example, a range of IP addresses extending from 192.168.0.1 to 192.168.0.128 can be allocated to the hosts on that network:

```
range 192.168.0.5 192.168.0.128;
```

You should also define your lease times, both a default and a maximum:

```
default-lease-time 21600;
max-lease-time 43200;
```

For a small, simple home network, you just need to list the **range** declaration along with any global options, as shown here. If your DHCP server is managing several subnetworks, you will have to use **subnet** declarations.

To assign dynamic addresses to a network, the DHCP server will require that your network topology be mapped. This means it needs to know what network addresses belong to a given network. Even if you use only one network, you will need to specify the address space for it. You define a network with the **subnet** declaration. Within this **subnet** declaration, you can specify any parameters, declarations, or options to use for that network. The **subnet** declaration informs the DHCP server of the possible IP addresses encompassed by a given subnet. This is determined by the network IP address and the

netmask for that network. The next example defines a local network with address spaces from 192.168.0.0 to 192.168.0.255. The **range** declaration allows addresses to be allocated from 192.168.0.5 to 192.168.0.128.

```
subnet 192.168.1.0 netmask 255.255.255.0 {
          range 192.168.0.5 192.168.0.128;
 }
```

Versions of DHCP prior to 3.0 required that you even map connected network interfaces that are not being served by DHCP. Thus, each network interface has to have a corresponding **subnet** declaration. Those not being serviced by DHCP don't have a **not authoritative** parameter as shown here (192.168.2.0 being a network not to be serviced by DHCP). In version 3.0 and later, DHCP simply ignores unmapped network interfaces:

```
subnet 192.168.2.0 netmask 255.255.255.0 {
      not authoritative;
}
```

The implementation of a very simple DHCP server for dynamic addresses is shown in the sample **dhcpd.conf** file that follows:

```
/etc/dhcpd.conf
option routers 192.168.0.1;
option subnet-mask 255.255.255.0;
option domain-name "mytrek.com";
option domain-name-servers 192.168.0.1;

subnet 192.168.1.0 netmask 255.255.255.0 {
         range 192.168.0.5 192.168.0.128;
         default-lease-time 21600;
         max-lease-time 43200;
         }
```

DHCP Dynamic DNS Updates

For networks that also support a Domain Name Server, dynamic allocation of IP addresses currently needs to address one major constraint: DHCP needs to sync with a DNS server. A DNS server associates hostnames with particular IP addresses, whereas in the case of dynamic allocation, the DHCP server randomly assigns its own IP addresses to different hosts. These may or may not be the same as the IP addresses that the DNS server expects to associate with a hostname. A solution to this problem is being developed, called Dynamic DNS. With Dynamic DNS, the DHCP server is able to automatically update the DNS server with the IP addresses the DHCP server has assigned to different hosts.

NOTE *Alternatively, if you want to statically synchronize your DHCP and DNS servers with fixed addresses, you configure DHCP to assign those fixed addresses to hosts. You can then have the DHCP server perform a DNS lookup to obtain the IP address it should assign, or you can manually assign the same IP address in the DHCP configuration file. Performing a DNS lookup has the advantage of specifying the IP address in one place, the DNS server.*

The DHCP server has the ability to dynamically update BIND DNS server zone configuration files. You enable dynamic updates on a DNS server for a zone file by specifying the **allow-update** option for it in the **named.conf** file. Furthermore, it is strongly encouraged that you use TSIG signature keys to reference and authenticate the BIND and DHCP servers. Currently, DHCP uses the Interim DNS Update Scheme to perform dynamic DNS updates, replacing an earlier Ad-Hoc DNS Update Scheme. A finalized version will be implemented in future DHCP releases. You can find detailed information about dynamic DNS in the **dhcpd .conf** Man page.

Enabling the use of a TSIG key involves syncing configurations for both your DHCP and DNS servers. Both have to be configured to use the same key for the same domains. First you need to create a shared secret TSIG signature key using **dnssec-keygen**. In the DNS server, you place TSIG key declarations and **allow-update** entries in the server's **named.conf** file, as shown in this example:

```
key mydhcpserver {
algorithm HMAC-MD5;
secret "ONQAfbBLnvWU9H8hRqq/WA==";
};


zone "mytrek.com" {
       type master;
       file "mytrek.com";
       allow-update {key mydhcpserver;};
 };

zone "1.168.192.IN-ADDR.ARPA" {
       type master;
       file "192.168.0";
       allow-update {key mydhcpserver;};
};
```

In the DHCP server, you place a corresponding TSIG key declaration and **allow-update** entries in the server's **dhcpd.conf** file, as shown in this example. The **key** declaration has the same syntax as the DNS server. DHCP **zone** statements are then used to specify the IP address of the domain and the TSIG key to use. The domain names and IP addresses need to match exactly in the configuration files for both the DNS and DHCP servers. Unlike in the **named.conf** file, there are no quotes around the domain name or IP addresses in the **dhcpd.conf** file. In the **dhcpd.conf** file, the domain names and IP addresses used in the **zone** statement also need to end with a period, as they do in the DNS zone files. The **key** statement lists the key to use. Though the DHCP server will try to determine the DNS servers to update, it is recommended that you explicitly identify them with a primary statement in a **zone** entry.

```
key mydhcpserver {
    algorithm HMAC-MD5;
    secret "ONQAfbBLnvWU9H8hRqq/WA==";
    };
```

```
zone mytrek.com. {                #DNS domain zone to update
       primary 192.168.0.1;       #address of DNS server
       key mydhcpserver;         #TSIG signature key
 };

zone 1.168.192.IN-ADDR.ARPA. {    #domain PTR zone to update
       primary 192.168.0.1;       #address of DNS server
       key mydhcpserver;          # TSIG signature key
};
```

To generate a fully qualified hostname to use in a DNS update, the DHCP server will normally use its own domain name and the hostname provided by a DHCP client (see the **dhcpd.conf** Man page for exceptions). Should you want to assign a specific hostname to a host, you can use the **ddns-hostname** statement to specify it in the host's hardware section. The domain name is specified in the **domain-name** option:

```
option domain-name "mytrek.com"
```

The DNS update capability can be turned on or off for all domains with the **ddns-update-style** statement. It is on by default. To turn off DNS updates for particular domains, you can use the **ddns-updates** statement. This is also on by default.

A simple DNS update configuration for a DHCP server in the **dhcpd.conf** file is shown here:

```
/etc/dhcpd.conf
option routers 192.168.0.1;
 option subnet-mask 255.255.255.0;
 option domain-name "mytrek.com";
 option domain-name-servers 192.168.0.1;
 key mydhcpserver {
         algorithm HMAC-MD5;
         secret "ONQAfbBLnvWU9H8hRqq/WA==";
         };

subnet 192.168.1.0 netmask 255.255.255.0 {
         range 192.168.0.5 192.168.0.128;
         default-lease-time 21600;
         max-lease-time 43200;
         zone mytrek.com. {
                  primary 192.168.0.1;
                  key mydhcpserver;
                  }
         zone 1.168.192.IN-ADDR.ARPA. {
                  primary 192.168.0.1;
                  key mydhcpserver;
                  }
}
```

DHCP Subnetworks

If you are dividing your network space into several subnetworks, you can use a single DHCP server to manage them. In that case, you will have a **subnet** declaration for each subnetwork. If you are setting up your own small network, you use a network address

beginning with 192.168. The range specifies possible IP addresses within that network so, for a network with the address 192.168.0.0, you create a **subnet** declaration with the netmask 255.255.255.0. Within this declaration, you place a **range** declaration along with any other information you want to give to your client hosts. In the following example, a range of IP addresses extending from 192.168.0.1 to 192.168.0.75 can be allocated to the hosts on that network:

```
subnet 192.168.0.0 netmask 255.255.255.0 {
 range 192.168.0.5 192.168.0.75;
}
```

You may want to specify different policies for each subnetwork, such as different lease times. Any entries in a **subnet** declaration will override global settings. So if you already have a global lease time set, a lease setting in a **subnet** declaration will override it for that subnet. The next example sets different lease times for different subnets, as well as different address allocations. The lease times for the first subnet are taken from the global lease time settings, whereas the second subnet defines its own lease times:

```
default-lease-time 21600;
max-lease-time 43200;

subnet 192.168.1.0 netmask 255.255.255.0 {
      range 192.168.0.5 192.168.0.75;
      }
subnet 192.168.1.128 netmask 255.255.255.252 {
      range 192.168.0.129 192.168.0.215;
      default-lease-time 56000;
      max-lease-time 62000;
       }
```

If your subnetworks are part of the same physical network, you need to inform the server of this fact by declaring them shared networks. You do this by placing subnet declarations within a **shared-network** declaration, specifying the shared network's name. The name can be any descriptive name, though you can use the domain name. Any options specified within the **shared-network** declaration and outside the subnet declarations will be global to those subnets. In the next example, the subnets are part of the same physical network and so are placed within a **shared-network** declaration:

```
shared-network mytrek.com
{
default-lease-time 21600;
max-lease-time 43200;
subnet 192.168.1.0 netmask 255.255.255.0 {
        range 192.168.0.5 192.168.0.75;
        }
subnet 192.168.1.128 netmask 255.255.255.252 {
        range 192.168.0.129 192.168.0.215;
        default-lease-time 56000;
        max-lease-time 62000;
        }
}
```

NOTE *Within a network, you can have some subnets that run DHCP servers and some that don't. In that case, you can use the DHCP Relay Agent to let DHCP clients on a subnet without a DHCP server use a DHCP server running on another subnet. The DHCP Relay Agent,* ***dhcrelay****, can be managed with the* ***service*** *command. It is configured with the* ***/etc/sysconfig/dhcrelay*** *file, where you can specify the network interfaces to receive requests from and the DHCP servers to use.*

DHCP Fixed Addresses

Instead of using a pool of possible IP addresses for your hosts, you may want to give each one a specific addresses. Using the DHCP server still gives you control over which address will be assigned to a given host. However, to assign an address to a particular host, you need to know the hardware address for that host's network interface card (NIC). In effect, you have to inform the DHCP server that it has to associate a particular network connection device with a specified IP address. To do that, the DHCP server needs to know which network device you are referring to. You can identify a network device by its hardware address, known as its MAC address. To find out a client's hardware address, you log in to the client and use the **ifconfig** command to find out information about your network devices. To list all network devices, use the **-a** option. If you know your network device name, you can use that. The next example will list all information about the first Ethernet device, **eth0**:

```
ifconfig eth0
```

This will list information on all the client's network connection devices. The entry (usually the first) with the term **HWaddr** will display the MAC address. Once you have the MAC address, you can use it on the DHCP server to assign a specific IP address to that device.

In the **dhcpd.conf** file, you use a **host** declaration to set up a fixed address for a client. Within the **host** declaration, you place a **hardware** option in which you list the type of network connection device and its MAC address. Then you use the **fixed-address** parameter to specify the IP address to be assigned to that device. In the following example, the client's network device with a MAC address of 08:00:2b:4c:29:32 is given the IP address 192.168.0.2:

```
host rabbit {
             option host-name "rabbit.mytrek.com"
             hardware ethernet 08:00:2b:4c:29:32;
             fixed-address 192.168.0.2;
             }
```

You can also have the DHCP server perform a DNS lookup to obtain the host's IP address. This has the advantage of letting you manage IP addresses in only one place, the DNS server. Of course, this requires that the DNS server be operating so that the DHCP server can determine the IP address. For example, a proxy server connection (which can provide direct web access) needs just an IP address, not a DNS hostname, to operate. If the DNS server were down, the preceding example would still assign an IP address to the host, whereas the example on the following page would not.

```
host rabbit {
             option host-name "rabbit.mytrek.com"
             hardware ethernet 08:00:2b:4c:29:32;
             fixed-address rabbit.mytrek.com;
             }
```

You can also use the **host** declaration to define network information for a diskless workstation or terminal. In this case, you add a **filename** parameter specifying the boot file to use for that workstation or terminal. Here the terminal called **myterm** obtains boot information from the server **turtle.mytrek.com**:

```
host myterm {
             option host-name "myterm.mytrek.com"
             filename "/boot/vmlinuz";
             hardware ethernet 08:00:2b:4c:29:32;
             server-name "turtle.mytrek.com";
             }
```

The implementation of a very simple DHCP server for fixed addresses is shown in the sample **dhcpd.conf** file that follows. In the second **host** declaration, the DHCP will perform a DNS lookup to obtain the IP address of **rabbit.mytrek.com**:

```
/etc/dhcpd.conf
option routers 192.168.0.1;
option subnet-mask 255.255.255.0;
option domain-name "mytrek.com";
option domain-name-servers 192.168.1.1;

subnet 192.168.1.0 netmask 255.255.255.0 {
host turtle {
                  option host-name "turtle.mytrek.com"
                  hardware ethernet 08:00:2b:4c:29:32;
                  fixed-address 192.168.0.1;
                  }
host rabbit {
                  option host-name "rabbit.mytrek.com"
                  hardware ethernet 00:80:AD:30:17:2A;
                  fixed-address rabbit.mytrek.com;
                  }
host lizard {
                  option host-name "lizard.mytrek.com"
                  hardware ethernet 00:70:2b:4b:29:14;
                  fixed-address 192.168.0.3;
                  }
}
```

A **host** declaration can also be placed within the **subnet** declaration to provide information about particular hosts in that subnet.

A common candidate for a fixed address is the DNS server for a network. Usually, you want the DNS server located at the same IP address, so that it can be directly accessed. The DHCP server can then provide this IP address to its clients.

36 CHAPTER

NFS and NIS

Linux provides several tools for accessing files on remote systems connected to a network. The Network File System (NFS) enables you to connect to and directly access resources such as files or devices like CD-ROMs that reside on another machine. The new version, NFS4, provides greater security, with access allowed by your firewall. The Network Information Service (NIS) maintains configuration files for all systems on a network.

Network File Systems: NFS and /etc/exports

NFS enables you to mount a file system on a remote computer as if it were local to your own system. You can then directly access any of the files on that remote file system. This has the advantage of allowing different systems on a network to access the same files directly, without each having to keep its own copy. Only one copy will be on a remote file system, which each computer can then access. You can find out more about NFS at its website at **nfs.sourceforge.net**.

NOTE *Mac OS X for Macintosh computers, which is based on BSD Unix, now supports NFS for file sharing. To access Apple file systems and printers using older Apple operating systems, you can use Netatalk, which implements the classic AppleTalk and AppleShare IP network protocols on Unix and Linux systems. The current Netatalk website is* ***netatalk.sourceforge.net****, with links to the FAQ and the HOWTO sections.*

NFSv4

NFS version 4 is a new version of the NFS protocol with enhanced features like greater security, reliability, and speed. Most of the commands are the same with a few changes. For example, when you mount an NFSv4 file system, you need to specify the **nfs4** file type. Also, you mount the entire NFSv4 file system at one location. To remount specific directories to different locations (NFS format), use the **bind option**.

```
/home/richlp            *(fsid=0,ro,sync)
```

The preceding entry lets you then mount the file system to the **/home/richlp** directory without having to specify it in the mount operation.

```
# mount -t nfs4  rabbit.mytrek.com:/  /home/dylan/projects
```

NFS Daemons

NFS operates over a TCP/IP network. The remote computer that holds the file system makes it available to other computers on the network. It does so by exporting the file system, which entails making entries in an NFS configuration file called **/etc/exports**, as well as by running several daemons to support access by other systems. These include **rpc.mountd**, **rpc.nfsd**, and **rpc.portmapper**. Access to your NFS server can be controlled by the **/etc/hosts.allow** and **/etc/hosts.deny** files. The NFS daemons are listed here:

- **rpc.nfsd** Receives NFS requests from remote systems and translates them into requests for the local system.
- **rpc.mountd** Performs requested mount and unmount operations.
- **rpc.portmapper** Maps remote requests to the appropriate NFS daemon.
- **rpc.rquotad** Provides user disk quota management.
- **rpc.statd** Provides locking services when a remote host reboots.
- **rpc.lockd** Handles lock recovery for systems that have gone down.

NOTE *It is advisable to use NFS on a local secure network only. If used over the Internet, NFS opens your system up to nonsecure access.*

Starting and Stopping NFS

The **nfs** service script will start up the **portmapper**, **nfsd**, **mountd**, and **rquotad** daemons. To enable NFS locking, you use the **nfslock** script. This will start up the **statd** and **lockd** daemons. NFS locking provides for better recovery from interrupted operations that can occur from system crashes on remote hosts. You can use **chkconfig** on Red Hat, Fedora, SUSE and similar distributions, or **services-admin** or **sysv-rc-conf** on Debian, Ubuntu, and other distributions to have NFS start up automatically.

To see if NFS is actually running, you can use the **rpcinfo** command with the **-p** option. You should see entries for **mountd** and **nfs**. If not, NFS is not running.

NFS Configuration: /etc/exports

An entry in the **/etc/exports** file specifies the file system to be exported and the hosts on the network that can access it. For the file system, enter its *mountpoint,* the directory to which it was mounted on the host system. This is followed by a list of hosts that can access this file system along with options to control that access. A comma-separated list of export options placed within a set of parentheses may follow each host. For example, you might want to give one host read-only access and another read and write access. If the options are preceded by an * symbol, they are applied to any host. A list of options is provided in Table 36-1. The format of an entry in the **/etc/exports** file is shown here:

```
directory-pathname   host-designation(options)
```

NFS Host Entries

You can have several host entries for the same directory, each with access to that directory:

```
directory-pathname   host(options) host(options)  host(options)
```

General Option	Description
`secure`	Requires that requests originate on secure ports, those less than 1024. This is on by default.
`insecure`	Turns off the `secure` option.
`ro`	Allows only read-only access. This is the default.
`rw`	Allows read/write access.
`sync`	Performs all writes when requested. This is the default.
`async`	Performs all writes when the server is ready.
`no_wdelay`	Performs writes immediately, not checking to see if they are related.
`wdelay`	Checks to see if writes are related, and if so, waits to perform them together. Can degrade performance. This is the default.
`hide`	Automatically hides an exported directory that is the subdirectory of another exported directory. The subdirectory has to be explicitly mounted to be accessed. Mounting the parent directory does not allow access. This is the default.
`no_hide`	Does not hide an exported directory that is the subdirectory of another exported directory (opposite of `hide`). Only works for single hosts and can be unreliable.
`subtree_check`	Checks parent directories in a file system to validate an exported subdirectory. This is the default.
`no_subtree_check`	Does not check parent directories in a file system to validate an exported subdirectory.
`insecure_locks`	Does not require authentication of locking requests. Used for older NFS versions.
User ID Mapping	**Description**
`all_squash`	Maps all UIDs and GIDs to the anonymous user. Useful for NFS-exported public FTP directories, news spool directories, and so forth.
`no_all_squash`	The opposite option to `all_squash`. This is the default setting.
`root_squash`	Maps requests from remote root user to the anonymous UID/GID. This is the default.
`no_root_squash`	Turns off root squashing. Allows the root user to access as the remote root.
`anonuid` `anongid`	Sets explicitly the UID and GID of the anonymous account used for `all_squash` and `root_squash` options. The defaults are nobody and nogroup.

Table 36-1 The /etc/exports Options

You have a great deal of flexibility when specifying hosts. For hosts within your domain, you can just use the hostname, whereas for those outside, you need to use a fully qualified domain name. You can also use just the host's IP address. Instead of a single host, you can reference all the hosts within a specific domain, allowing access by an entire network. A simple way to do this is to use the ***** for the host segment, followed by the domain name for the network, such as ***.mytrek.com** for all the hosts in the **mytrek.com** network. Instead of domain names, you can use IP network addresses with a CNDR format where you specify the netmask to indicate a range of IP addresses. You can also use an NIS netgroup name to reference a collection of hosts. The NIS netgroup name is preceded by an **@** sign.

```
directory       host(options)
directory       *(options)
directory       *.domain(options)
directory       192.168.1.0/255.255.255.0(options)
directory       @netgroup(options)
```

NFS Options

Options in **/etc/exports** operate as permissions to control access to exported directories. Read-only access is set with the **ro** option, and read/write with the **rw** option. The **sync** and **async** options specify whether a write operation is performed immediately (**sync**) or when the server is ready to handle it (**async**). By default, write requests are checked to see if they are related, and if so, they are written together (**wdelay**). This can degrade performance. You can override this default with **no_wdelay** and have writes executed as they are requested. If two directories are exported, where one is the subdirectory of another, the subdirectory is not accessible unless it is explicitly mounted (**hide**). In other words, mounting the parent directory does not make the subdirectory accessible. The subdirectory remains hidden until also mounted. You can overcome this restriction with the **no_hide** option (though this can cause problems with some file systems). If an exported directory is actually a subdirectory in a larger file system, its parent directories are checked to make sure that the subdirectory is the valid directory (**subtree_check**). This option works well with read-only file systems but can cause problems for write-enabled file systems, where filenames and directories can be changed. You can cancel this check with the **no_subtree_check** option.

NFS User-Level Access

Along with general options, there are also options that apply to user-level access. As a security measure, the client's root user is treated as an anonymous user by the NFS server. This is known as *squashing* the user. In the case of the client root user, squashing prevents the client from attempting to appear as the NFS server's root user. Should you want a particular client's root user to have root-level control over the NFS server, you can specify the **no_root_squash** option. To prevent any client user from attempting to appear as a user on the NFS server, you can classify them as anonymous users (the **all_squash** option). Such anonymous users only have access to directories and files that are part of the anonymous group.

Normally, if a user on a client system has a user account on the NFS server, that user can mount and access his or her files on the NFS server. However, NFS requires the user ID for the user be the same on both systems. If this is not the case, he or she is considered to be two different users. To overcome this problem, you can use an NIS service, maintaining user ID information in just one place, the NIS password file (see the following section for information on NIS).

NFS /etc/exports Example

Examples of entries in an **/etc/exports** file are shown here. Read-only access is given to all hosts to the file system mounted on the **/pub** directory, a common name used for public access. Users, however, are treated as anonymous users (**all_squash**). Read and write access is given to the **lizard.mytrek.com** computer for the file system mounted on the **/home/foodstuff** directory. The next entry allows access by **rabbit.mytrek.com** to the NFS server's CD-ROM, using only read access. The last entry allows anyone secure access to **/home/richlp**.

```
/etc/exports
/pub                    *(ro,insecure,all_squash,sync)
/home/foodstuff         lizard.mytrek.com(rw,sync)
/media/cdrom            rabbit.mytrek.com(ro,sync)
/home/richlp            *(secure,sync)
```

Applying Changes

Each time your system starts up the NFS server (usually when the system starts up), the **/etc/exports** file will be read and those directories specified will be exported. When a directory is exported, an entry for it is made in the **/var/lib/nfs/xtab** file. It is this file that NFS reads and uses to perform the actual exports. Entries are read from **/etc/exports** and corresponding entries made in **/var/lib/nfs/xtab**. The **xtab** file maintains the list of actual exports.

If you want to export added entries in the **/etc/exports** file immediately, without rebooting, you can use the **exportfs** command with the **-a** option. It is helpful to add the **-v** option to display the actions that NFS is taking. Use the same options to effect any changes you make to the **/etc/exports** file.

```
exportfs -a -v
```

If you later make changes to the **/etc/exports** file, you can use the **-r** option to re-export its entries. The **-r** option will resync the **/var/lib/nfs/xtab** file with the **/etc/exports** entries, removing any other exports or any with different options.

```
exportfs -r -v
```

To both export added entries and re-export changed ones, you can combine the **-r** and **-a** options.

```
exportfs -r -a -v
```

Manually Exporting File Systems

You can also use the **exportfs** command to manually export file systems instead of using entries for them in the **/etc/exports** file. Export entries will be added to the **/var/lib/nfs/xtab** file directly. With the **-o** option, you can list various permissions and then follow them with the host and file system to export. The host and file system are separated by a colon. For example, to manually export the **/home/myprojects** directory to **golf.mytrek.com** with the permissions **ro** and **insecure**, you use the following:

```
exportfs -o rw,insecure golf.mytrek.com:/home/myprojects
```

You can also use **exportfs** to unexport a directory that has already been exported, either manually or by the **/etc/exports** file. Just use the **-u** option with the host and the directory exported. The entry for the export will be removed from the **/var/lib/nfs/xtab** file. The following example will unexport the **/home/foodstuff** directory that was exported to **lizard**
.mytrek.com:

```
exportfs -u lizard.mytrek.com:/home/foodstuff
```

NFS File and Directory Security with NFS4 Access Lists

With NFS4 you can set up access control lists (ACL) for particular directories and files. You use the NFS4 ACL tools to manage these lists (nfs4-acl-tools package). The NFS4 file system ACL tools include **nfs4_getfacl**, **nfs4_setfacl**, and **nfs4_editfacl**. Check the Man page for each for detailed options and examples. **nfs4_getfacl** will list the access controls for a specified file or directory. **nfs4_setfacl** will create access controls for a directory or file, and **nfs4_editfacl** will let you change them. **nfs4_editfacl** simply invokes **nfs_setfacl** with the **-e** option. When editing access controls, you are placed in an editor where you can make your changes. For setting access controls you can read from a file, the standard input, or list the control entries on the command line.

The file and directory access controls are more refined than the standard permissions described in Chapter 28. The ACL entries follow the syntax described in detail on the **nfs4_acl** Man page. An ACL entry begins with an entry type such as an accept or deny entry (**A** or **D**); followed by an ACL flag, which can specify group or inheritance capability and then the principal to which the ACL is applied; and finally, the list of access options, such as **r** for read or **w** for write. The principal is usually a user URL that is to be permitted or denied access. You can also specify groups, but you need to set the **g** group flag. The special URLs OWNER@, GROUP@, and EVERYONE@ correspond to the owner, group, and other access used on standard permissions. The following example provides full access to the owner, but gives only read and execute access to the user **george@rabbit.com**. Group write and execute access is denied.

```
A::OWNER@:rwadtTnNcCy
A::george@rabbit.com:rxtncy
D:g:GROUP@:waxtc
```

In addition to read, write, and execute permissions (**r**,**w**,**x**), ACL lists also provide attribute reads (**t**,**n**) and attribute writes (**T**,**N**), as well as ACL read (**c**) and write (**C**) access. NFS read and write synchronization is enabled with the **y** option. The ability to delete files and directories is provided by the **d** option and for subdirectories with the **D** option. The **a** option lets you append data and create subdirectories. Keep in mind that **rtncy** are all read options, whereas **wadDTNC** are write options, while **x** remains the execute option. You will need **y** for any synchronized access. The **C** option in particular is very powerful as it allows the user to change the access controls (lowercase **c** allows only reading of the access controls).

Controlling Access to NFS Servers

You can use several methods to control access to your NFS servers, such as using hosts.allow and hosts.deny to permit or deny access, as well as using your firewall to intercept access.

/etc/hosts.allow and /etc/hosts.deny

The **/etc/hosts.allow** and **/etc/hosts.deny** files are used to restrict access to services provided by your server to hosts on your network or on the Internet (if accessible). For example, you can use the **hosts.allow** file to permit access by certain hosts to your FTP server. Entries in the **hosts.deny** file explicitly deny access to certain hosts. For NFS, you can provide the same kind of security by controlling access to specific NFS daemons.

NOTE *You can further secure your NFS transmissions by having them operate over TCP instead of UDP. Use the* **`tcp`** *option to mount your NFS file systems (UDP is the default). However, performance does degrade for NFS when it uses TCP.*

Portmapper Service

The first line of defense is to control access to the portmapper service. The portmapper tells hosts where the NFS services can be found on the system. Restricting access does not allow a remote host to even locate NFS. For a strong level of security, you should deny access to all hosts except those that are explicitly allowed. In the **hosts.deny** file, you place the following entry, denying access to all hosts by default. ALL is a special keyword denoting all hosts.

```
portmap:ALL
```

In the **hosts.allow** file, you then enter the hosts on your network, or any others that you want to permit to have access to your NFS server. Again, you specify the portmapper service and then list the IP addresses of the hosts to which you are permitting access. You can list specific IP addresses, or a network range using a netmask. The following example allows access only by hosts in the local network, 192.168.0.0, and by the host 10.0.0.43. You can separate addresses with commas:

```
portmap: 192.168.0.0/255.255.255.0, 10.0.0.43
```

The portmapper is also used by other services such as NIS. If you close all access to the portmapper in **hosts.deny**, you will also need to allow access to NIS services in **hosts.allow**, if you are running them. These include ypbind and ypserver. In addition, you may have to add entries for remote commands like **`ruptime`** and **`rusers`**, if you are supporting them.

It is also advisable to add the same level of control for specific NFS services. In the **hosts.deny** file, you add entries for each service, as shown here:

```
mountd:ALL
rquotad:ALL
statd:ALL
lockd:ALL
```

Then, in the **hosts.allow** file, you can add entries for each service:

```
mountd:  192.168.0.0/255.255.255.0, 10.0.0.43
rquotad: 192.168.0.0/255.255.255.0, 10.0.0.43
statd:   192.168.0.0/255.255.255.0, 10.0.0.43
lockd:   192.168.0.0/255.255.255.0, 10.0.0.43
```

Netfilter Rules

You can further control access using Netfilter to check transmissions from certain hosts on the ports used by NFS services. (See Chapter 20 for an explanation of Netfilter.) The portmapper uses port 111, and nfsd uses 2049. Netfilter is helpful if you have a private network that has an Internet connection and you want to protect it from the Internet. Usually a specific network device, such as an Ethernet card, is dedicated to the Internet connection. The following examples assume that device **eth1** is connected to the Internet. Any packets attempting access on port 111 or 2049 are refused.

```
iptables -A INPUT -i eth1 -p 111 -j DENY
iptables -A INPUT -i eth1 -p 2049 -j DENY
```

To enable NFS for your local network, you will have to allow packet fragments. Assuming that **eth0** is the device used for the local network, you could use the following example:

```
iptables -A INPUT -i eth0 -f -j ACCEPT
```

Mounting NFS File Systems: NFS Clients

Once NFS makes directories available to different hosts, those hosts can then mount those directories on their own systems and access them. The host needs to be able to operate as an NFS client. Current Linux kernels all have NFS client capability built in. This means that any NFS client can mount a remote NFS directory that it has access to by performing a simple mount operation.

Mounting NFS Automatically: /etc/fstab

You can mount an NFS directory either by an entry in the **/etc/fstab** file or by an explicit **mount** command. You have your NFS file systems mounted automatically by placing entries for them in the **/etc/fstab** file. An NFS entry in the **/etc/fstab** file has a mount type of NFS. An NFS file system name consists of the hostname of the computer it is located on, followed by the pathname of the directory where it is mounted. The two are separated by a colon. For example, **rabbit.trek.com:/home/project** specifies a file system mounted at **/home/project** on the **rabbit.trek.com** computer. The format for an NFS entry in the **/etc/fstab** file follows. Notice that the file type is **nfs**.

```
host:remote-directory    local-directory      nfs   options   0   0
```

You can also include several NFS-specific mount options with your NFS entry. You can specify the size of datagrams sent back and forth and the amount of time your computer waits for a response from the host system. You can also specify whether a file system is to be hard-mounted or soft-mounted. For a *hard-mounted* file system, your computer continually tries to make contact if for some reason the remote system fails to respond. A *soft-mounted* file system, after a specified interval, gives up trying to make contact and issues an error message. A hard mount is the default. A system making a hard-mount attempt that continues to fail will stop responding to user input as it tries continually to achieve the mount. For this reason, soft mounts may be preferable, as they will simply stop attempting a mount that continually fails. Table 36-2 and the Man pages for **mount** contain a listing of these NFS client options. They differ from the NFS server options indicated previously.

Option	Description
rsize=*n*	The number of bytes NFS uses when reading files from an NFS server. The default is 1024 bytes. A size of 8192 can greatly improve performance.
wsize=*n*	The number of bytes NFS uses when writing files to an NFS server. The default is 1024 bytes. A size of 8192 can greatly improve performance.
timeo=*n*	The value in tenths of a second before sending the first retransmission after a timeout. The default value is seven-tenths of a second.
retry=*n*	The number of minutes to retry an NFS mount operation before giving up. The default is 10,000 minutes (one week).
retrans=*n*	The number of retransmissions or minor timeouts for an NFS mount operation before a major timeout (default is 3). At that time, the connection is canceled or a "server not responding" message is displayed.
soft	Mount system using soft mount.
hard	Mount system using hard mount. This is the default.
intr	Allows NFS to interrupt the file operation and return to the calling program. The default is not to allow file operations to be interrupted.
bg	If the first mount attempt times out, continues trying the mount in the background. The default is to fail without backgrounding.
tcp	Mounts the NFS file system using the TCP protocol, instead of the default UDP protocol.

TABLE 36-2 NFS Mount Options

An example of an NFS entry follows. The remote system is **rabbit.mytrek.com**, and the file system is mounted on **/home/projects**. This file system is to be mounted on the local system as the **/home/dylan/projects** directory. The **/home/dylan/projects** directory must already be created on the local system. The type of system is NFS, and the **timeo** option specifies the local system waits up to 20 tenths of a second (two seconds) for a response. The mount is a soft mount and can be interrupted by NFS.

```
rabbit.mytrek.com:/home/projects /home/dylan/projects  nfs
soft,intr,timeo=20
```

Mounting NFS Manually: mount

You can also use the **mount** command with the **-t nfs** option to mount an NFS file system explicitly. For an NFSv4 file system you use **-t nfs4**. To mount the previous entry explicitly, use the following command:

```
# mount -t nfs -o soft,intr,timeo=20   \
          rabbit.mytrek.com:/home/projects   /home/dylan/projects
```

You can, of course, unmount an NFS directory with the **umount** command. You can specify either the local mountpoint or the remote host and directory, as shown here:

```
umount /home/dylan/projects
umount  rabbit.mytrek.com:/home/projects
```

Mounting NFS on Demand: autofs

You can also mount NFS file systems using the automount service, autofs. This requires added configuration on the client's part. The autofs service will mount a file system only when you try to access it. A directory change operation (`cd`) to a specified directory will trigger the mount operation, mounting the remote file system at that time.

The autofs service is configured using a master file to list map files, which in turn lists the file systems to be mounted. The **/etc/auto.master** file is the autofs master file. The master file will list the root pathnames where file systems can be mounted along with a map file for each of those pathnames. The map file will then list a key (subdirectory), mount options, and the file systems that can be mounted in that root pathname directory. On some distributions, the **/auto** directory is already implemented as the root pathname for file systems automatically mounted. You can add your own file systems in the **/etc/auto.master** file along with your own map files, if you wish. You will find that the **/etc/auto.master** file contains the following entry for the **/auto** directory, listing **auto.misc** as its map file:

```
/auto    auto.misc   --timeout 60
```

Following the map file, you can add options, as shown in the preceding example. The `timeout` option specifies the number of seconds of inactivity to wait before trying to automatically unmount.

In the map file, you list the key, the mount options, and the file system to be mounted. The key will be the subdirectory on the local system where the file system is mounted. For example, to mount the **/home/projects** directory on the **rabbit.mytrek.com** host to the **/auto/projects** directory, you use the following entry:

```
projects  soft,intr,timeo=20   rabbit.mytrek.com:/home/projects
```

You can also create a new entry in the master file for an NFS file system, as shown here:

```
/myprojects    auto.myprojects   --timeout 60
```

You then create an **/etc/auto.myprojects** file and place entries in it for NFS file system mounts, like the following:

```
dylan    soft,intr,rw   rabbit.mytrek.com:/home/projects
newgame  soft,intr,ro   lizard.mytrek.com:/home/supergame
```

Network Information Service: NIS

On networks supporting NFS, many resources and devices are shared by the same systems. Normally, each system needs its own configuration files for each device or resource. Changes entail updating each system individually. However, NFS provides a special service called the Network Information Service (NIS) that maintains such configuration files for the entire network. For changes, you need only to update the NIS files. NIS works for information required for most administrative tasks, such as those relating to users, network access, or devices. For example, you can maintain user and password information with an NIS service, having only to update those NIS password files.

NOTE *NIS+ is a more advanced form of NIS that provides support for encryption and authentication. However, it is more difficult to administer.*

NIS was developed by Sun Microsystems and was originally known as Sun's Yellow Pages (YP). NIS files are kept on an NIS server (NIS servers are still sometimes referred to as YP servers). Individual systems on a network use NIS clients to make requests from the NIS server. The NIS server maintains its information on special database files called *maps*. Linux versions exist for both NIS clients and servers. Linux NIS clients easily connect to any network using NIS.

NOTE *Instead of NIS, many networks now use LDAP to manage user information and authentication.*

The NIS client is installed as part of the initial installation on most Linux distributions. NIS client programs are ypbind (the NIS client daemon), ypwhich, ypcat, yppoll, ypmatch, yppasswd, and ypset. Each has its own Man page with details of its use. The NIS server programs are ypserv (the NIS server), ypinit, yppasswdd, yppush, ypxfr, and netgroup—each also with its own Man page.

NIS Servers

You have significant flexibility when setting up NIS servers. If you have a small network, you may need only one NIS domain, for which you would have one NIS server. For larger networks, you can divide your network into several NIS domains, each with its own server. Even if you only have one domain, you may want several NIS slave servers. For an NIS domain, you can have a master NIS server and several NIS slave servers. The slave servers can act as backups, in case the master server goes down. A slave server only contains copies of the configuration files set up on the NIS master server.

Configuring an NIS server involves several steps:

1. Define the NIS domain name that the NIS server will work for.
2. Start the **ypserv** daemon.
3. In the **/var/yp/Makefile** file, set any NIS server options and specify the configuration files to manage.
4. Use **`/usr/lib/ypinit`** to create the NIS versions of the configuration files.

Defining the NIS Domain

You first have to define an NIS domain name. You can have the NIS domain defined whenever you start up your system by defining the NIS_DOMAIN variable in the **/etc/sysconfig/network** file. To this variable, you assign the name you want to give your NIS domain. The following example defines the NIS domain called **myturtles.nis**:

```
NIS_DOMAIN=myturtles.nis
```

When first setting up the server, you may want to define your NIS domain name without having to restart your system. You can do so with the **`domainname`** command, as shown here:

```
domainname myturtles.nis
```

NIS server options are kept in the **/etc/ypserv.conf** file. Check the Man page for that file for details.

Setting NIS Server Options

Next, edit the **/var/yp/Makefile** file to select the configuration files that the NIS server will maintain and set any NIS server options. Standard options, as well as most commonly used configuration files, are usually already set up.

NIS server options are listed first. The **NOPUSH** option will be set to true, indicating that there are no slave NIS servers. If you are setting up any slave NIS servers for this domain, you will have to set this option to false:

```
NOPUSH = true
```

The minimum user and group IDs are set to 500. These are set using the **MINUID** and **MINGID** variables:

```
MINUID=500
MINGID=500
```

Most distributions use a shadow password and shadow group files to encrypt passwords and groups; the **MERGE_PASSWD** and **MERGE_GROUP** settings will be set to true. NIS will merge shadow password information into its password file:

```
MERGE_PASSWD=true
MERGE_GROUP=true
```

The directories where NIS will find password and other configuration files are then defined using the **YPSRCDIR** and **YPPWDIR** variables. Normally, the **/etc** directory holds your configuration files:

```
YPSRCDIR = /etc
YPPWDDIR = /etc
```

Then the configuration files that NIS can manage are listed. Here, you will find entries like **PASSWD** for password, **GROUP** for your groups, and **PRINTCAP** for your printers. A sample of the entries is shown here:

```
GROUP       = $(YPPWDDIR)/group
PASSWD      = $(YPPWDDIR)/passwd
SHADOW      = $(YPPWDDIR)/shadow
GSHADOW     = $(YPPWDDIR)/gshadow
ALIASES     = /etc/aliases
ETHERS      = $(YPSRCDIR)/ethers     # ethernet addresses (for rarpd)
BOOTPARAMS  = $(YPSRCDIR)/bootparams # for booting Sun boxes (bootparamd)
HOSTS       = $(YPSRCDIR)/hosts
NETWORKS    = $(YPSRCDIR)/networks
PRINTCAP    = $(YPSRCDIR)/printcap
PROTOCOLS   = $(YPSRCDIR)/protocols
```

Specifying Shared Files

The actual files that are shared on the network are listed in the **all:** entry, which follows the list of configuration files. Only some of the files defined are listed as shared, those listed in the first line after **all:**. The remaining lines are automatically commented out

(with a preceding **#** sign). You can add files by removing the **#** sign or moving their entries to the first line.

```
all:  passwd group hosts rpc services netid protocols mail \
      # netgrp shadow publickey networks ethers bootparams printcap \
      # amd.home auto.master auto.home auto.local passwd.adjunct \
      # timezone locale netmasks
```

Be sure not to touch the remainder of the Makefile.

Creating the NIS Database

You then enter the **ypinit** command with the **-m** option to create the NIS database consisting of the NIS configuration files. Your NIS server will be detected, and you will be asked to enter the names of any slave NIS servers used on this NIS domain. If there are any, enter them. When you are finished, press CTRL-D. The NIS database files are then created. The ypinit command is located in the **/usr/lib/yp** directory.

```
ypinit -m
```

For an NIS slave server, you use

```
ypinit -s masterhost
```

Should you receive the following error, it most likely means that your NIS server was not running. Be sure to start **ypserv** before you run **ypinit**.

```
failed to send 'clear' to local ypserv: RPC: Program not registeredUpdating
```

If you later need to update your NIS server files, you change to the **/var/yp** directory and issue the **make** command.

```
cd /var/yp
make
```

Controlling Access

The **/var/yp/securenets** file enables access by hosts to your NIS server. Hosts can be referenced by network or individually. Entries consist of a subnet mask and an IP address. For example, you can give access to all the hosts in a local network with the following entry:

```
255.255.255.0  192.168.1.0
```

For individual hosts, you can use the mask 255.255.255.255 or just the term "host," as shown here:

```
host   192.168.1.4
```

Controlling how different hosts access NIS shared data is determined in **/etc/ypserv.conf**.

Netgroups

You can use NIS to set up netgroups, which allow you to create network-level groups of users. Whereas normal groups are created locally on separate hosts, an NIS netgroup can be used for networkwide services. For example, you can use NIS netgroups to control access to NFS file systems. Netgroups are defined in the **/etc/netgroup** file. Entries consist of a netgroup name followed by member identifiers consisting of three segments: the host, the user, and the NIS domain:

```
group    (host, user, NIS-domain) (host, user, NIS-domain) ...
```

For example, in the NIS domain **myturtles.nis**, to define a group called **myprojects** that consists of the user **chris** on the host **rabbit** and the user **george** on the host **lizard.mytrek.com**, you use the following:

```
myprojects (rabbit, chris, myturtles.nis) \
                          (lizard.mytrek.com, george, myturtles.nis)
```

A blank segment will match on any value. The following entry includes all users on the host **rabbit**:

```
newgame (rabbit,,myturtles.ni)
```

If your use of a group doesn't need either a user or a host segment, you can eliminate one or the other using a hyphen (**-**). The following example generates a netgroup consisting just of hostnames, with no usernames:

```
myservers (rabbit,-,) (turtle.mytrek.com,-,)
```

You can then reference different netgroups in various configuration files by prefixing the netgroup name with an **@** sign, as shown here:

```
@newgame
```

NIS Clients

For a host to use NIS on your network, you first need to specify your NIS domain name on that host. In addition, your NIS clients need to know the name of your NIS server. If you installed Linux on a network already running NIS, you may have already entered this information during the installation process. You can specify your NIS domain name and server in the **/etc/yp.conf** file.

Accessing the Server

Each NIS client host on your network then has to run the ypbind NIS client to access the server. In the client's **/etc/yp.conf** file, you need to specify the NIS server it will use. The following entry references the NIS server at 192.168.1.1:

```
ypserver 192.168.1.1
```

Alternatively, you can specify the NIS domain name and the server it uses:

```
domain mydomain.nis  server servername
```

The following entry for domain and server would be in **/etc/yp.conf** for the **myturtle.nis** NIS domain using the **turtle.mytrek.com** server:

```
domain myturtles.nis server turtle.mytrek.com
```

You can use **ypcat** to list any of the NIS configuration files. The **ypwhich** command will display the name of the NIS server your client is using. **ypmatch** can be used to find a particular entry in a configuration file.

```
ypmatch cecelia passwd.
```

Users can change their passwords in the NIS **passwd** file by using the **yppasswd** command; it works the same as the **passwd** command. You will also have to have the **yppasswdd** daemon running.

Specifying Configuration Files with nsswitch.conf

To ensure that the client accesses the NIS server for a particular configuration file, you should specify **nisplus** in that file's entry in the **/etc/nsswitch.conf** file. The **nisplus** option refers to the NIS version 3. The **nis** option refers to the older NIS version 2. The **/etc/nsswitch.conf** file specifies where a host should look for certain kinds of information. For example, the following entry says to check the local configuration files (**files**) first and then the NIS server (**nisplus**) for password data:

```
passwd:   files nisplus
```

The **files** designation says to first use the system's own files, those on the localhost. **nis** says to look up entries in the NIS files, accessing the NIS server. **nisplus** says to use NIS+ files maintained by the NIS+ server. **dns** says to perform DNS lookups; it can only be used on files like **hosts** that contain hostnames. These are some standard entries:

```
passwd:     files nisplus
shadow:     files nisplus
group:      files nisplus

hosts:      files nisplus dns
bootparams: nisplus [NOTFOUND=return] files

ethers:     files
netmasks:   files
networks:   files
protocols:  files nisplus
rpc:        files
services:   files nisplus
netgroup:   files nisplus
publickey:  nisplus
automount:  files nisplus
aliases:    files nisplus
```

This page intentionally left blank

CHAPTER 37

Distributed Network File Systems

For very large distributed systems like Linux clusters, Linux also supports distributed network file systems, such as Coda, Intermezzo, Red Hat Global File System (GFS and GFS 2), and the Parallel Virtual File System (PVFS2). These systems build on the basic concept of NFS as well as RAID techniques to create a file system implemented on multiple hosts across a large network, in effect distributing the same file system among different hosts at a very low level (see Table 37-1). You can think of it as a kind of RAID array implemented across network hosts instead of just a single system. Instead of each host relying on its own file systems on its own hard drive, they all share the same distributed file system that uses hard drives collected on different distributed servers. This provides far more efficient use of storage available to the hosts, as well as providing for more centralized management of file system use.

Parallel Virtual File System (PVFS)

The Parallel Virtual File System (PVFS) implements a distributed network file system using a management server that manages the files system on different I/O servers. Management servers maintain the file system information, including access permissions, directory structure, and metadata information. Requests for access to a file are submitted by a client of the management server. The management server then sets up a connection between the client and the I/O servers that hold the requested file's data, and access operations such as read and write tasks are carried out directly between the client and the I/O servers. PVFS can be implemented transparently using a kernel module to use the kernel's virtual file system. The PVFS file system can then be mounted by a client like any file system. In a PVFS implementation, the file system is organized into stripes of data, similar to a RAID array, but the stripes are distributed to different hosts on the network. Files are accessed as a collection of stripes that can be distributed across this network.

A new version of PVFS, known as PVFS2, is currently available from **pvfs.org**. You can download the source code from there. PVFS is a joint project with the Parallel Architecture Research Laboratory at Clemson University and the Mathematics and Computer Science Division at Argonne National Laboratory.

Website	Name
fedoraproject.org/wiki/Tools/GFS	Fedora GFS resources and links
pvfs.org	Parallel Virtual File System, PVFS2 (open source)
sourceware.org/gfs	Global File System (Fedora and commercial versions)
coda.cs.cmu.edu **ftp.coda.cs.cmu.edu/pub/coda/linux**	Coda File system, disconnected mobile access (experimental)
inter-mezzo.org	Intermezzo (open source)
clusterfs.com	Lustre

TABLE 37-1 Distributed File Systems

The PVFS manager server uses two configuration files, **pvfs2-fs.conf** and **pvfs2-server .conf**. You create these files using a configuration script, **`pvfs2-genconfig`**. The script will prompt you for configuration information such as the protocol, port, log file, and storage directory. To configure a cluster network, be sure to enter the hosts that will operate as I/O servers.

The management server is **pvfs2-server**. To manage the server, use the **`pvfs2-server`** service script.

On I/O servers, you run the **`pvfs2-server`** script to create storage space, and then start the server. The I/O servers use the **/etc/pvfs2-fs.conf** configuration file. You can use the **`pvfs2-server`** server script to turn the server on automatically.

On the clients that will use the PVFS system, you need to install file system configuration information and access tools. Each client will need a PVFS daemon, **pvfs2-client** and its supporting module and library. File systems mount configurations can be held in **/etc/pvf2stab**, though you can also place entries directly in the **/etc/fstab** file. To mount a PVFS file system, you can also use the **`pvfs2`** type in the **`mount`** command with the **`-t`** option. PVFS2 provides is own set of file commands, such as **`pvfs2-ping`**, **`pvfs2-cp`**, and **`pvfs2-ls`**, to list and access files on its file system.

A PVFS2 file system can be accessed by the client either through a kernel module to provide direct Linux file system access, or with the MPI-IO interface, which requires recompiling using the libpvfs2 library but provides better performance for parallel applications. Parallel support in PVFS2 is implemented with the message passing interface (MPI) as supported by ROMIO (**www-unix.mcs.anl.gov/romio**) and MPICH2 (**www-unix.mcs.anl.gov/mpi/mpich2**) available from the MCS Division of the Argonne National Laboratory.

NOTE *For applications to take full advantage of PVFS, they should be PVFS-enabled, which will stripe their file data for better use on PVFS systems.*

Coda

Coda was developed by Carnegie Mellon University as an experimental project, though freely available. Some of its features include support for mobile computing, access controls, and bandwidth adaptation. You can obtain information about Coda from **coda.cs.cmu.edu** and download packages for some releases from **ftp.coda.cs.cmu.edu/pub/coda/linux**.

Using a kernel module to interface with the virtual file system, distributed Coda files can be accessed from the Coda directory on a client, usually **/coda**. Coda will maintain a cache of frequently accessed files on the client to improve efficiency. The cache is maintained by a cache manager called venus that handles all file system requests. The use of a cache allows for a disconnected operation on a file, letting users work on a file locally and then update it later with the main servers. A disconnected operation works well for mobile computing, where laptops may be disconnected from the network for periods of time. Corresponding databases for frequently used files by users, known as hoards, are also maintained on the server to facilitate updates.

To configure clients, you will need the coda-debug-client package. Use the **venus-setup** script to configure the client, and then start up Coda with the **venus** daemon. For the server, you will need to install the coda-dbug-server package and run **venus-setup** to configure your server.

Red Hat Global File System (GFS and GFS 2)

Red Hat has released its Global File System (GFS) as an open source, freely available distributed network file system. The original GFS version was included with Fedora 4 and 5. Starting with Fedora Core 6, the new version of GFS, GFS 2, is included, using a similar set of configuration and management tools, as well as native kernel support. Instead of a variety of seemingly unrelated packages, GFS 2 is implemented with just three: **gfs2-utils**, **cman**, and **lvm-cluster**. Native kernel support for GFS 2 provides much of the kernel-level operations.

A distributed network file system builds on the basic concept of NFS as well as RAID techniques to create a file system implemented on multiple hosts across a large network, in effect distributing the same file system among different hosts at a very low level. You can think of it as a kind of RAID array implemented across network hosts instead of just a single system. That is, instead of each host relying on its own file systems on its own hard drive, they all share the same distributed file system that uses hard drives collected on different distributed servers. This provides far greater efficient use of storage available to the hosts and provides for more centralized management of file system use. GFS can be run either directly connected to a SAN (storage area network) or using GNBD (Global Network Block Device) storage connected over a LAN. The best performance is obtained from a SAN connection, whereas a GNBD format can be implemented easily using the storage on LAN (Ethernet)–connected systems. As with RAID devices, mirroring, failover, and redundancy can help protect and recover data.

GFS separates the physical implementation from the logical format. A GFS appears as a set of logical volumes on one seamless logical device that can be mounted easily to any directory on your Linux file system. The logical volumes are created and managed by the Cluster Logical Volume Manager (CLVM), which is a cluster-enabled LVM. Physically, the file system is constructed from different storage resources, known as cluster nodes, distributed across your network. The administrator manages these nodes, providing needed mirroring or storage expansion. Should a node fail, GFS can fence a system off until it has recovered the node. Setting up a GFS requires planning. You have to determine ahead of time different settings like the number and names of your Global File Systems, the nodes that will be able to mount the file systems, fencing methods, and the partitions and disks to use.

For detailed information check the Cluster Project Page site at **sourceware.org/cluster**. Listed are the packages used in both GFS (Cluster Components—Old) and GFS 2 (Cluster Components—New). Here you will find links for documentation like the clustering FAQ. The Red Hat GFS Administrators Guide can be helpful but may be dated. The guide can be found on the Red Hat documentation page located at **redhat.com**. (Bear in mind that GFS now uses logical volumes instead of pools to set up physical volumes.)

Website	Name
redhat.com/software/rha/gfs	Global File System (Red Hat commercial version)
redhat.com/docs/manuals/csgfs/	Global File System Red Hat manuals (Red Hat Enterprise implementation)
sourceware.org/cluster	Cluster Project website, which includes links for GFS documentation
/etc/cluster.conf	GFS cluster configuration file (css)

GFS 2 Packages (Fedora Core 6 and On)

The original GFS, GFS 1, used a variety of separate packages for cluster servers and management tools, which did not appear related just by their names. With GFS 2, these packages have been combined into the **cman** and **gfs2-tools** packages. Here you will find tools such as fence, cman cluster manager, dlm locking control, and **ccs** cluster configuration. Cluster configuration is supported by the Cluster Configuration System, **ccs**. Fencing is used to isolate failed resources. It is supported by in the fence server. LVM cluster support is located in a separate package like **lvm2-cluster** (Fedora) or **clvm** (Debian).

To run a cluster, you need both a cluster manager and locking mechanism. **cman** with the Distributed Lock Manager (**dlm**) implements cluster management and locking. **cman** manages connections between cluster devices and services, using **dlm** to provide locking. The **dlm** locking mechanism operates as a daemon with supporting libraries.

All these services are invoked by the `cman` script, which checks the **/etc/cluster.conf** file for cluster configuration.

GFS 2 Service Scripts

To start the GFS file system, you run the `cman` script to start the needed daemon and implement your configuration. The `cman` script will run the **ccsd** to start up configuration detection, **fenced** for fencing support, **dlm_controld** for cluster management **dlm** locking, and **cman** for cluster management. The script will check for any GFS configuration settings in the **/etc/sysconfig/cluster** file. You use the `gfs2` script to mount your GFS 2 file systems. To shut down the GFS file system service, you use the `cman` script with the `stop` option.

```
service cman start
service gfs2 start
```

The `gfs2` service script will mount GFS file systems to the locations specified in the **/etd/fstab** file. You will need entries for all the GFS file systems you want to mount in **/etc/fstab**. The stop option will unmount the file systems. You can use `cman_tool` to add a node to a cluster or remove a node from the cluster.

Implementing a GFS 2 File System

To set up a GFS 2 file system, you first need to create cluster devices using the physical volumes and organize them into logical volumes. You use the CLVM (Clustering Logical Volume Manager) to set up logical volumes from physical partitions (in the past you used a volume manager called pool to do this). You can then install GFS file systems on these logical volumes directly. CLVM operates like LVM, using the same commands. It works over a distributed network and requires that the **clvmd** server be running.

You then configure your system with the Cluster Configuration System. Create a **/etc/cluster.conf** file and set up your configuration. The configuration will include information like the nodes used, the fencing methods, and the locking method used. Consult the **cluster.conf** Man page for configuration details. Test the configuration with the **ccs_test** tool.

```
ccs_test mygfs
```

You then use the **ccs_tool** to create **cluster.ccs**, **fence.ccs**, and **node.ccs** configuration files. These files are organized into a CCS archive that is placed on each node and cluster device.

On each node, start the **ccsd** configuration, the **fenced** fencing server, and the locking method you want to use, such as **dlm**. Check the respective Man pages for details on the locking servers. You can start the servers with their service scripts, as noted previously.

To create new file systems on the cluster devices, you use the **gfs2_mkfs** command and mount them with the **-t gfs2** option. The following command creates a GFS file system on the **/dev/gv0/mgfs** and then mounts it to the **/mygfs** directory. For **gfs2_mkfs**, the **-t** option indicates the lock table used and the **-p** option specifies the lock protocol. The **-j** option specifies the number of journals.

```
gfs2_mkfs  -t mycluster:mygfs  -p lock_dlm  -j 2   /dev/vg0/mgfs
mount -t gfs /dev/vg0/mgfs /gfs1
```

To have the **gfs** service script mount the GFS file system for you, you need to place an entry for it in the **/etc/fstab** file. If you do not want the file system automatically mounted, add the **noauto** option.

```
/dev/vg0/mgfs   /mygfs    gfs2   noauto,defaults   0   0
```

With GFS **/etc/fstab** entries, you can then use the **gfs2** script to mount the GFS file system.

```
service gfs start
```

GFS Tools

GFS has several commands in different categories, such as those that deal with fencing, like **fence_tool**; **gulm_tool** to manage gulm locking; and those used for configuration, like **cman_tool**. The GFS commands for managing GFS file systems are listed in Table 37-2. Check their respective Man pages for detailed descriptions.

Command	Description
ccs	CCS service script to start Cluster Configuration Service server
ccs_tool	CCS configuration update tool
ccs_test	CCS diagnostic tool to test CCS configuration files
ccsd	Daemon run on nodes to provide CCS configuration data to cluster software
clvmd	Cluster Logical Volume Manager daemon, needed to create and manage LVM cluster devices, also a service script to start **clvmd**
cman	The Cluster Manager, **cman**, startup script, uses dlm for locking (**cman** is run as a kernel module directly)
cman_tool	Manages cluster nodes, requires **cman**
dlm	Distributed Lock Manager, implemented as a kernel module, invoked by the **cman** script
fence	Fence overview
fenced	Fencing daemon, also a service script for starting the fenced daemon
fence_tool	Manages the fenced daemon
fence_node	Invokes a fence agent
fencing agents	Numerous fencing agents available for different kinds of connections, see **fence** Man page
fence_manual	Fence agent for manual interaction
fence_ack_manual	User interface for fence_manual
gfs2	GFS 2 service script to mount GFS 2 file systems, also a Man page overview
gfs2_mount	Invoked by mount; use **-t gfs2** mount option
gfs2_fsck	The GFS 2 file system checker
gfs2_grow	Grows a GFS 2 file system
gfs2_jadd	Adds a journal to a GFS 2 file system
mkfs.gfs2	Makes a GFS 2 file system
gfs2_quota	Manipulates GFS 2 disk quotas
gfs2_tool	Manages a GFS 2 file system
getfacl	Gets the ACL permissions for a file or directory
setfacl	Sets access control (ACL) for a file or directory
rmanager	Resource Group Manager, manage user services

Table 37-2 GFS Tools, Daemons, and Service Scripts

GFS File System Operations

Several GFS commands manage the file system, such as **gfs2_mount** for mounting file systems, **gfs2_mkfs** to make a GFS file system, **gfs2_fsck** to check and repair, and **gfs2_grow** to expand a file system. Check their respective Man pages for detailed descriptions.

NOTE *For GFS 1, you use the same names for the GFS tools without the number 2; that is,* ***gfs*** *instead of* ***gfs2****.*

To mount a GFS file system, you use the **mount** command specifying **gfs2** as the mount type, as in

```
mount  -t gfs2  /dev/vg0/mgfs  /mygfs
```

This will invoke the **gfs2_mount** tool to perform the mount operation. Several GFS-specific mount options are also available, specified with the **-o** option, such as **lockproto** to specify a different lock protocol and **acl** to enable ACL support.

To check the status of a file system, you can use **gfs2_fsck**. This tool operates much like fsck, checking for corrupt systems and attempting repairs. You must first unmount the file system before you can use **gfs2_fsck** on it.

Should you add available space to the device on which a GFS file system resides, you can use **gfs2_grow** to expand the file system to that available space. It can be run on just one node to expand the entire cluster. If you want journaling, you first have to add journal files with the **gfs2_jadd** tool. **gfs2_grow** can only be run on a mounted GFS file system.

Journal files for GFS are installed in space outside of the GFS file system, but on the same device. After creating a GFS file system, you can run **gfs2_add** to add the journal files for it. If you are expanding a current GFS file system, you need to run **gfs2_add** first. Like **gfs2_grow**, **gfs2_add** can only be run on mounted file systems. With the **setfacl** command you can set permissions for files and directories.

As noted previously, to create a GFS file system you use the **gfs2_mkfs** command. The **-t** option specifies the lock table to use, the **-j** option indicates the number of journals to create, and the **-p** option specifies the lock protocol to use.

The Resource Group Manager, **rgmanager**, provides a command line interface for managing user services and resources on a GFS file system, letting you perform basic administrative tasks like setting user quotas, shutting down the system (**clushutdown**), and getting statistics on GFS use (**clustat**). The primary administrative tool is **clusterfs**. Options can be set in the **/etc/sysconfig/cluster** file. You start up **rgmanager** with the **rgmanager** script. This starts up the **clurgmgrd** daemon, providing access to the GFS system.

GFS also supports access controls. You can restrict access by users or groups to certain files or directories, specifying read or write permissions. With the **setfacl** command you can set permissions for files and directories. You use the **-m** option to modify an ACL permission and **-x** to delete it. The **getfacl** obtains the current permissions for files or directories. The following sets read access by the user **dylan** to **myfile**.

```
setfacl -m u:dylan:r myfile
```

GFS 1

GFS 1, the first version of GFS, implements GFS with a series of separate packages, seemingly unrelated. To use GFS 1, you have to install a number of different packages that include the GFS 1 tools, the locking method you want to use, and configuration tools. The GFS 1 tools and locking methods also have several corresponding kernel module and header packages. See **sourceware.org/cluster** under the heading "Cluster Components—Old" for a listing of GFS 1 tools. There were kernel module packages for the different types of kernels: i586, i686, SMP, and Xen.

Appendix

Where to Obtain Linux Distributions

Linux distributions provide professional-level and very stable Linux systems along with the KDE and GNOME GUI interfaces, flexible and easy-to-use system configuration tools, an extensive set of Internet servers, a variety of different multimedia applications, and thousands of Linux applications of all kinds. You can find recent information about different distributions at their respective websites. From there you can also download DVD or CD install images that you can burn and use to install a particular distribution. Several distributions also provide Live-CDs that you can use to run Linux using just a CD-ROM drive, without having to install Linux on your hard drive.

Most software from the major distributions is available for download from the distribution-supported repositories. Install disk images are available also, either as smaller desktop-only installs, or larger server installs. You can usually use a Live-CD, if provided, to install Linux. Distribution strategy relies on install disks with a selected collection of software that can be later updated and enhanced from the very large collection of software on the distribution repository. This means that the collection of software in an initial installation can be relatively small. Software on the repository is also continually updated, so any installation will likely have to undergo extensive updates from the repository.

Be sure to first read the install guide for the distribution you are installing. These are provided online at the distribution website and can be read using any browser.

Several popular Linux distributions and their websites are listed in Table A-1. In most cases you only need to search the name on a search service like Google to find the site. The DistroWatch site provides a detailed listing and description of most distributions.

```
www.distrowatch.com
```

Name	Site
Debian	**debian.org**
Fedora	**fedoraproject.org**
openSUSE	**opensuse.com**
Red Hat	**redhat.com**
Ubuntu	**ubuntu.com**
Gentoo	**gentoo.org**
CentoOS	**centos.org**
MEPIS	**mepis.org**
Slackware	**slackware.com**
Turbolinux	**turbolinux.com**

Table A-1 Popular Linux Distributions

Index

Symbols

! (history command)
    editing in C shell, 61
    referencing events with, 41–42, 44, 59–60
    using, 40–41
    using in Z shell, 63
! operator, using with eth0 device, 383
!~ operator, using with TCSH shell, 82
!= operator, using with TCSH shell, 82
(pound) symbol
    as prompt, 21, 36
    using with ServerName directive, 452
    using with service script tags, 413–414
    using with xinetd services, 409–410
$ (dollar sign)
    appearance before shells, 36
    function of, 21, 68
    preceding csh command with, 57
    as prompt, 99
    representing words with, 60
    using with BASH automatic completions, 39
$ operator, using with shell variables, 68
% (percent) symbol
    preceding job numbers with, 54
    using with Apache log files, 457–458
& (ampersand) operator
    function of, 68
    as shell symbol, 44
    using with background jobs, 53–54
&& command
    executing commands with, 37
    using with kernel, 682
* (asterisk)
    function of, 68
    matching characters with, 45
    as shell symbol, 44
    use in syslogd daemon, 540
    using with C shell history, 60
    using with .Xresources file, 159
. (dot) command, re-executing .bash_profile script with, 103
. (dot) files, use of, 116
. (period)
    representing directories with, 68, 90
    using with extensions, 116
.. symbol, using with cd command, 124
/ (root directory)
    contents of, 118
    indicating, 120
    role in FHS, 584–585
/ (slash), use with BASH command line, 39
: (colon), using in Z shell, 63
; (semicolon), separating commands with, 37, 44
? (question mark)
    function of, 68
    matching single characters with, 45–46
    as shell symbol, 44
@ (at sign), using with BASH automatic completions, 39
[] (brackets)
    function of, 68
    matching character ranges with, 46
    as shell symbol, execution of, 44
    using with test operations, 78–79
\ (backslash)
    displaying, 100
    executing as shell symbol, 44
    using with commands, 36
    using to quote characters, 46–47
    using with exclamation point (!), 60
    using with prompts, 99
^ (caret)
    representing words with, 60
    using in C shell, 57

` (back quotes)
versus single quotes (‘), 70
using with commands, 70
{} (curly braces), using with patterns, 47
| (pipe operator)
connecting commands with, 51
as shell symbol, 44
using with standard output, 49
|& shell symbol, execution of, 44
~ (tilde), using with BASH automatic completions, 39
< (less than) character
function of, 68
as shell symbol, 44
= (assignment operator)
using with xinetd services, 418
using with shell variables, 68
=~ operator, using with TCSH shell, 82
== operator, using with TCSH shell, 82
> (greater than) character
function of, 68
redirecting standard output with, 48–50
as shell symbol, 44
>! (overwriting) shell symbol, execution of, 44
>& (redirection) shell symbols, execution of, 44
>> (redirection operator)
as shell symbol, execution of, 44
using with standard output, 50, 52
2> shell symbol, execution of, 44
2>>, using to append standard error, 52
20-storage-methods.fdi file, contents of, 656
50-udev.nodes file, contents of, 657–658
‘ (single quotes)
versus back quotes (`), 69
using with commands, 69
./, preceding configure command with, 232
“ “ (double quotes), using with characters, 47, 68–69

A

.a extension, meaning of, 233
AbiWord word processor, features of, 243
absolute pathname, explanation of, 119
absolute permissions, using, 566–568
accept and reject CUPS command line tools, features of, 512
ACCEPT policy, inclusion in myfilter script, 393
ACCEPT target, using with packets, 379–380, 383
access.db file, using with Sendmail, 497
accounts, accessing on other systems, 309
ACLs (access control lists)
creating for Squid, 470
setting up with NFS4, 766
AddEncoding directive, using with Apache web server, 455
AddHandler directive
using with Apache web server, 455
using with SSIs (server-side includes), 462
AddLanguage directive, using with Apache web server, 455
addprincipal command, using with kadmin tool, 372
AddType directive
using with Apache web server, 455
using with SSIs (server-side includes), 462
Advanced Intrusion Detection Environment (AIDE), implementing intrusion detection with, 325
Advanced Linux Sound Architecture (ALSA), website for, 260–261
Advanced Package Tool (APT), using with Debian, 228
agetty program, using, 661
AH (authentication header), applying instructions to, 351
AIDE (Advanced Intrusion Detection Environment), implementing intrusion detection with, 325
.aiff web file type, description of, 283
AIM (AOL Instant Messenger), features of, 305
alias command
issuing, 91–93
using with TCSH shell, 107–108
Alias directive, using with Apache web server, 453
aliases
creating for commands, 69
support in Sendmail, 485, 487
allow directive, using with Apache web server, 453
allow keyword, using in SELinux, 339–340
AllowOverride directive, using with Apache web server, 453
ALSA (Advanced Linux Sound Architecture), website for, 260–261
ALT key. *See also* keyboard shortcuts
using in BASH automatic completion, 39
using with history lists, 41

Amanda, backing up hosts with, 695–698
amdump command, making backups with, 697–698
ampersand (&) operator
    function of, 68
    as shell symbol, 44
    using with background jobs, 53–54
anacron, using with cron, 531
AND, logical operator for, 78
Anonymous directive, using with Apache web server, 456
anonymous FTP
    allowing, 297, 425–427
    explanation of, 425
    server directories for, 426–427
anonymous login, performing for FTP sites, 423
anycast address, explanation of, 721
AOL Instant Messenger (AIM), features of, 305
Apache Configuration Tool, features of, 463–464
Apache installations on Linux, features of, 446
Apache Jakarta Project, features of, 445–446
Apache web server
    AddEncoding directive used with, 455
    AddHandler directive used with, 455
    AddLanguage directive used with, 455
    AddType directive used with, 455
    Alias directive used with, 453
    allow directive used with, 453
    AllowOverride directive used with, 453
    Anonymous directive used with, 456
    AuthConfig directive used with, 453
    AuthDBGroupFile directive used with, 457
    AuthDBMGGroupFile directive used with, 457
    AuthDBMGUserFile directive used with, 457
    AuthDBUserFile directive used with, 457
    authentication in, 456–457
    AuthGroupFile directive used with, 456–457
    AuthName directive used with, 456
    AuthType directive used with, 456
    AuthUserfile directive used with, 456–457
    and automatic directory indexing, 455
    and CGI (Common Gateway Interface) files, 455
    configuration files for, 448
    CustomLog directive used with, 457–458
    DefaultType directive used with, 455
    deny directive used with, 453
    directives used with, 448–449
    Directory directive used with, 452
    DirectoryIndex directive used with, 454
    directory-level configuration of, 452–453
    dynamic virtual hosting in, 459–462
    FancyIndexing directive used with, 455
    features of, 444–445
    FileInfo directive used with, 453
    FormatLog directive used with, 457
    global configuration of, 449–451
    and IP-based virtual hosting, 459, 461
    KeepAlive directive used with, 449
    KeepAliveRequests directive used with, 449–450
    Limit directive used with, 453
    Listen directive used with, 450, 459
    LoadModule directive used with, 450
    log files in, 457–458
    LogFormat directive used with, 457
    MaxClients directive used with, 451
    MaxRequestsPerChild directive used with, 451
    and MIME types, 454–455
    modules for, 450
    MPM configuration of, 450–451
    name-based virtual hosting in, 459
    Options directive used with, 453
    Redirect directive used with, 454
    ServerAdmin directive used with, 451
    ServerLimit directive used with, 451
    ServerName directive used with, 451–452
    ServerRoot directive used with, 449
    ServerTokens directive used with, 449
    starting and stopping, 447–448
    StartServer directive used with, 451
    ThreadsPerChild directive used with, 451
    Timeout directive used with, 449
    TransferLog directive used with, 458
    TypesConfig directive used with, 455
    URL for, 12
    and URL pathnames, 453–454
    use of MPMs with, 447
    UseCanonicalName directive used with, 460–461
    using apxs application with, 450

Apache web server (*Continued*)
virtual hosting on, 458–462
VirtualDocumentRoot directive used with, 460
VirtualScriptAlias directive used with, 460
Apache-SSL, description of, 444
apol Policy Analysis tool, using in SELinux, 334
applets. *See also* GNOME applets
features of, 23–24
for KDE panel, 206
application documentation, accessing, 29
application launchers
using with GNOME panel objects, 189–190
using with Nautilus, 184
applications, making available to users, 234
APT (Advanced Package Tool), using with Debian, 228
apxs application, using with Apache web server, 450
archive contents, displaying with tar, 134
archived tar files, compressing, 139
archives. *See also* files
checking contents of, 230
compressing, 137–138
creating on tape, 138
creating with tar, 134–136
extracting contents of, 139
extracting with tar, 136
updating with tar utility, 136–137
archiving to floppies, 137
arguments. *See also* script arguments
aliasing, 92
parsing on command line, 68
running commands as, 37
using in commands, 28
argv array, using, 72–73
ARRAY entries, generating for RAID, 637
AS (authentication server), role in Kerberos, 370
ASCII default, using with FTP clients, 295–296
assignment operator (=)
using with xinetd services, 418
using with shell variables, 68
asterisk (*)
function of, 68
matching characters with, 45
use in syslogd daemon, 540
using with C shell history, 60
using with .Xresources file, 159
asterisk (*) shell symbol, execution of, 44
at (@ sign), using with BASH automatic completions, 39
AT&T Unix Korn shell. *See* Korn shell
ATTRS key, using with udevinfo command, 651
.au web file type, description of, 283
audio applications, support for, 260–261
audit2allow command, using in SELinux, 334–335
auditd server, function of, 541–542
Auditing System, features of, 541–542
AUTH command, using with SASL, 496
AuthConfig directive, using with Apache web server, 453
AuthDBGroupFile directive, using with Apache web server, 457
AuthDBMGGroupFile directive, using with Apache web server, 457
AuthDBMGUserFile directive, using with Apache web server, 457
AuthDBUserFile directive, using with Apache web server, 457
authentication
in Apache web server, 456–457
function of, 373
in Kerberos, 369–371
in OpenSSH, 361
in ProFTPD (Professional FTP Daemon), 436
authentication header (AH), applying instructions to, 351
authentication server (AS), role in Kerberos, 369–370
AuthGroupFile directive, using with Apache web server, 456–457
AuthName directive, using with Apache web server, 456
AuthType directive, using with Apache web server, 456
AuthUserfile directive, using with Apache web server, 456–457
autofs service, mounting NFS on demand with, 770
automatic completions, examples of, 39
.avi web file type, description of, 283
azap tool, tuning TV channels with, 263–264

B

back quotes (`)
versus single quotes (’), 70
using with commands, 70

background
placing commands in, 53–54
running jobs in, 53–54
backing up files, 81
backslash (\)
displaying, 100
executing as shell symbol, 44
using with commands, 36
using to quote characters, 46–47
using with ! (exclamation point), 60
using with prompts, 99
backup devices, restoring from, 703
BackupPC tool, features of, 694–695
backups
with dump utility, 698–701
of files and directories, 693–694
of hosts with Amanda, 695–698
recording with dump utility, 700
recovering, 701–703
using archive tools with, 693–694
Balsa mail client, using with GNOME, 270
BASH command line, automatic completion capabilities of, 38–39
BASH environment, specifying, 99
BASH shell. *See also* shells
accessing reference manual for, 37
command line editing in, 37
conditional control structures in, 78–80
configuration files for, 91
configuring, 105–106
control structures in, 79–80
controlling operations in, 93–94
features of, 89
history list in, 40
history utility in, 40–42
script capabilities of, 65–66
test operators in, 78
website for, 35
BASH_ENV variable, contents of, 99
.bash_logout file, features of, 106–107
.bash_profile script, 102–103
contents of, 101
editing, 102–103
function of, 554–555
placing PATH assignments in, 234–235
re-executing manually, 103
.bashrc file
configuring BASH shell with, 105–106
function of, 554–555
Bell Labs, popularity of Unix at, 7
Berkeley Software Distribution (BSD), development of, 7
Beryl, development of, 176
bg command, execution of, 53, 55
biff mail notification utility, features of, 273–274
/bin directory, contents of, 537, 586
bin directory, using with anonymous FTP, 426–427
.bin extension, meaning of, 220
binary default, using with FTP clients, 295–296
binary masks, using with absolute permissions, 566–568
binary method, using with ownership permissions, 569
Blackdown project, obtaining Linux ports of Java from, 287
BlankTime value, using in Xorg, 152
block device, explanation of, 657
/boot directory, kernels in, 674
boot disks, placing kernels on, 683
boot loader, using with kernel packages, 675–676
bootable RAID, configuring, 635
brackets ([])
function of, 68
matching character ranges with, 46
as shell symbol, 44
using with test operations, 78–79
break structure, using in TCSH shell, 86
breaksw keyword, using in TCSH shell, 82–83
broadcast addresses, function of, 719
BSD (Berkeley Software Distribution), development of, 7
bunzip2 command, decompressing files with, 140, 229
byte-stream format, using with files, 117
.bz2 extension, meaning of, 220
.bz2 files, decompressing, 229
bzImage file for kernels, installing manually, 682–683
bzip2 utility, compressing files with, 140

C

C shell. *See also* shells
accessing, 57
command line editing, 57
development of, 56–57
environment variables in, 76
pattern substitution command in, 57
referencing commands in, 59

C shell history utility
commands in, 57–58
defining with set command, 58
event editing, 61
event substitutions in, 59–60
pattern references used with, 58–59
using substitution command in, 61
caches, setting up in Squid, 473–474
callouts, relationship to HAL, 657
caret (^)
representing words with, 60
using in C shell, 57
case command, effect of, 78–80
cat command
issuing, 39, 50, 120–121
redirecting standard output of, 52
cbackup script, 81, 87
CD burners and rippers, availability of, 261
cd command
issuing, 118, 123
using with .. symbol, 124
CD device icons, displaying in KDE, 201
CD image files, creating, 611
CD readers, symbolic links used for, 648
cddrecord command, issuing, 612
CD/DVD devices, checking HAL links to, 261
cdpath variable, using in TCSH shell, 110
cdrecord application, using with DVDs, 610
CD-ROM discs
accessing from KDE desktop, 208
accessing with GNOME Volume Manager, 177
burning data to, 27
burning with GNOME Volume Manager, 177
mounting, 604–605
recording data to, 610–613
CD-ROM drives, device names for, 592
CD-R/RW discs, copying files to, 129
cedega, using to play Windows games on Linux, 31
Centros Linux, URL for, 5
certificates
use in SSL (Secure Sockets Layer), 465
using with IPsec, 355
CGI (Common Gateway Interface) files, use with Apache web server, 455
chage command
issuing, 556
options for, 557
chain rules, adding and modifying, 376–378
chains
defining user chains, 393
in Netfilter, 377
types of, 376
character device, explanation of, 657
characters
erasing, 21, 28
matching, 45–46
matching in filenames, 44
navigating, 37–38
quoting, 44, 46–47
specifying ranges of, 46
transposing, 37
chcon command, using in SELinux, 334
check-cdrom.sh script, function of, 648
checkmodule command, using in SELinux, 341, 344
checkpolicy command, using in SELinux, 344
chgrp command, issuing, 565
chkconfig command
configuring xinetd services for, 409–410
enabling and disabling xinetd services with, 409
enabling for xinetd services, 418
functionality of, 410
listing services with, 407–408
removing and adding services with, 409
starting and stopping services with, 408–409
using off option with, 408
using reset option with, 408
using with rsync, 428
chmod command
changing permissions with, 563
using with anonymous FTP, 426–427
using with group directories, 560
chown command
issuing, 565
using with anonymous FTP, 426
chroot operation, using with ProFTPD, 438
CIDR (classless interdomain routing), relationship to TCP/IP addresses, 714–717
class-based IP addressing, organization of, 712–713
classless interdomain routing (CIDR), relationship to TCP/IP addresses, 714–717
Cluster Project Page, website for, 780
cman script, running for GFS 2, 780
Coda, features of, 778–779
codecs, availability of, 258, 263
CodeWeavers, support for Wine, 32

colon (:), using in Z shell, 63
colors, listing in X Window System, 159
command arguments, relationship to shell variables, 68
command line
    accessing, 35
    accessing Linux from, 20–22
    executing, 44
    parsing arguments on, 68
    shutting down Linux from, 21–22
    starting GUI from, 24
    starting Linux from, 167–168
    switching to, 19, 160
    using, 27–28
command line editing
    availability in BASH shell, 37
    in C shell, 57
command line mail clients
    Mail utility, 272
    Mutt, 271
command line PPP access, implementing, 737–738
command line prompts, creating, 109
command output, piping, 44
commands
    aliasing, 92–93
    displaying last-executed commands, 28
    displaying prior to execution, 108
    displaying with history list, 40–43
    for editing, 38
    entering on command line, 36–37
    executing, 44
    executing on same line, 37
    as filters, 51
    format for, 28
    interrupting and stopping, 36
    listing recent use of, 40
    location of, 233
    looping, 77
    for movement, 38
    obtaining values from, 70
    performing history operations on, 61
    placing in background, 54
    quoting, 69
    referencing in C shell, 59
    referencing ranges for editing, 43
    running as arguments, 37
    sending data between, 50–51
    separating, 44
    tracking in BASH shell, 40–42
Common Gateway Interface (CGI) files, use with Apache web server, 455
Common Unix Printing System (CUPS). *See* CUPS (Common Unix Printing System)
Compiz, using with GNOME desktop, 176
completion commands, examples of, 40
compress command, using, 140
compressed archives. *See also* files
    downloading, 229
    identifying, 10
    installing software from, 228–233
    selecting install directory for, 230
compressing files
    with bzip2 utility, 140
    with gzip utility, 138–140
Concurrent Versions System (CVS), function of, 235
conditional control structures
    definition of, 77
    using, 78–80
config tool, starting, 678
configuration files
    functions of, 91
    listing, 90
    location of, 90, 537, 587
    user configuration files, 552
    using Makefiles with, 231
configure command
    options for, 232
    preceding with ./, 232
connection tracking
    specialization of, 384
    using, 383
connections, configuring with setkey, 351–353
context-sensitive help, availability of, 29
continue structure, using in TCSH shell, 86
control structures
    in TCSH and C shells, 81–88
    types of, 77
copying files, 126–129
Courier MTA (mail transfer agent), features of, 477
Courier-IMAP server, website for, 500
cp command
    entering on command line, 36–37
    issuing, 126
    meaning of, 124
    using -r option with, 130
    using with directories, 129–130
CPU kernel packages, availability of, 674
CREATE TABLE command, issuing, 517

cron daemon
directory names for, 531
editing in C shell history, 529
environment variables for, 528
using anacron with, 531
cron directory scripts, running, 530–531
cron service, scheduling tasks with, 527
crontab command, issuing, 529
crontab entries
fields for, 527–528
placing in cron.d directory, 528–529
CrossOver Office
availability of, 32
running, 238–239
website for, 238
cryptsetup command, issuing, 326
csh command, accessing C shell with, 57
CTRL key. *See also* keyboard shortcuts
using in BASH automatic completion, 39
using with history lists, 41
CUPS (Common Unix Printing System)
configuring, 508
configuring on GNOME, 505
configuring on KDE, 505
configuring remote printers on, 507
enabling on Samba, 507
features of, 503–504
CUPS command line administrative tools
accept and reject, 512
availability of, 510–511
enable and disable, 512
lpadmin, 511
lpinfo, 512
lpoptions, 511–512
CUPS command line print clients
lpc, 510
lpq and lpstat, 510
lpr, 509
lprm, 510
CUPS configuration tool, using, 506
CUPS directives, using, 508
CUPS printer classes, creating, 507
cupsd server, function of, 402
cupsd.conf file
contents of, 507
location of, 508
curl client, description of, 291
curl Internet client, features of, 293
curly braces ({}), using with patterns, 47
CustomLog directive, using with Apache web server, 457–458
CVS (Concurrent Versions System), function of, 235
Cyrus IMAP server, website for, 500

D

daemons
and FTP server software, 423
scripts related to, 402
shutting down, 412
startup and shutdown of, 403
database management systems
and SQL, 516–517
SQL databases, 245–247
Xbase databases, 248
database servers
availability of, 515
MySQL, 517–520
PostgreSQL, 520
database software, availability of, 11–12
date, prompt code for, 100
date command
description of, 525
executing, 37
issuing, 526–527
DB2 database, features of and website for, 12, 246–247
.deb extension, meaning of, 10, 220
Debian
features of, 227–228
runlevels for, 405
URL for, 5
Debian Package tool (dpkg), using, 228
DefaultType directive, using with Apache web server, 455
deny directive, using with Apache web server, 453
depmod command, detecting dependencies with, 666
Desktop directory in KDE, contents of, 215–216
desktop fonts, resizing, 25
desktops
configuring sessions in, 27
configuring with KDE, 23
starting, 24
/dev directories, contents of, 590
development libraries, using with compiled programs, 232
development resources, availability of, 13
device files
contents of, 639

creating with MAKEDEV command, 657–658
as symbolic links, 645–647
device information files
directories for, 656–657
HAL properties for, 654
device names, relationship to udev rules, 643–645
device-mapper, using with LVM, 621–622
devices
archiving, 134–138
backing up files to, 137
creating with mknod command, 659–660
identifying, 651
making accessible via HAL, 652
obtaining information about, 639–641
property entries for, 654–655
types of, 657
/dev/mapper directory, contents of, 621
/dev/pts entry, meaning of, 661
df command, issuing, 593–594
DHCP (Dynamic Host Configuration Protocol), dynamic IPv4 addresses for, 754–755
dhcp, enabling for Xen Virtual Machines, 691
DHCP dynamic DNS updates, implementing, 755–757
DHCP fixed addresses, using, 759–60
DHCP for IPv4, function of, 750
DHCP IPv4 client hosts, configuring, 750
DHCP IPv4 server, configuring, 751, 753–754
DHCP servers, running, 759
DHCP subnetworks, overview of, 757–758
DHCPv6, providing stateful autoconfiguration with, 748–749
Dia drawing program, availability of, 244
digiKam photo manager, features of, 256
digital signatures
checking in software packages, 323–325
combining encryption with, 315
use in SSL (Secure Sockets Layer), 465
using, 314
directories. *See also* parent directory
accessing, 118
backing up and restoring, 693–694
changing, 118
compressing with Zip utility, 140–141
configuration files in, 90
contents of, 117
copying and moving, 129–130
creating and deleting, 122–123
displaying absolute pathnames for, 122
displaying contents of, 123
erasing, 130
functions of, 120
locating, 126
managing, 121–124
navigating, 123
organization of files into, 8
ownership of, 563–565
permission operations for, 564
permissions assigned to, 561–563
recovering, 701–703
searching, 124–126
system directories in FHS, 585–587
Directory directive, using with Apache web server, 452
directory names, distinguishing from filenames, 123
directory permissions, setting and displaying, 568
DirectoryIndex directive, using with Apache web server, 454
disable and enable CUPS command line tools, features of, 512
disk quotas
and quota tools, 571
setting with edquota, 571–572
disk space usage, determining for file system, 593–594
disk usage, displaying, 544
disklist file, using with Amanda, 697
dispfile script, 75–76
dispfirst script, 73
displast script, 74
display managers
features of, 19–20
GDM (GNOME Display Manager), 164–166
KDM (KDE Display Manager), 166–167
and sessions, 162–163
types of, 160
using with X Window System, 160–162
XDM (X Display Manager), 163–164
distributed network file systems. *See also* file systems
Coda, 778–779
GFS (Global File System), 779–784
PVFS (Parallel Virtual File System), 777–778
distributed parity level of RAID, function of, 628

DivX for Linux
features of, 264
support for, 27
website for, 262
DLL files, using with Windows, 32
dmraid RAID tool, features of, 626–627
DNAT rules, explanation of, 385
DNAT target, using, 396–397
DNS (Domain Name Service)
and hosts.conf file, 726–727
and NSS (Name Service Switch), 727–729
overview of, 725–726
relationship to mail servers, 478–479
do loop control structure, function of, 80
document viewers, availability of, 244
documentation, availability of, 13–15
dollar sign ($)
appearance before shells, 36
function of, 21, 68
preceding csh command with, 57
as prompt, 21, 99
representing words with, 60
using with BASH automatic completions, 39
domain name server, explanation of, 719
Domain Name Service (DNS). *See* DNS (Domain Name Service)
domains, relationship to mail servers, 477
done keyword, explanation of, 81
DontZap value, using in Xorg, 152
DontZoom value, using in Xorg, 152
dot (.) command, re-executing .bash_profile script with, 103. *See also* period (.)
dot (.) files, use of, 116
double quotes (" "), using with characters, 47, 68–69
Dovecot, features of, 499
dpkg (Debian Package tool), using, 228
driver packages, downloading for module installation, 669
drives, replacing with LVM, 622–623
DROP policy
including in IPtables script, 389–390
specifying for FORWARD chain, 396
DROP target, using with packets, 379–380
dummy option, using with cdrecord command, 613
dump command, relationship to fstab file, 598
dump utility, making backups with, 698–701
DVB reception, support for, 263–264
DVD device icons, displaying in KDE, 201
DVD discs
accessing with GNOME Volume Manager, 177
burning data to, 27
burning with GNOME Volume Manager, 177
DVD distribution images, downloading with BitTorrent, 220–221
DVD players, accessing, 262–263
DVD players list, website for, 262
DVD readers, symbolic links used for, 648
DVD+RW tools, availability of, 613
dvdrecord command, issuing, 612
DVD::rip
features of, 263
website for, 262
DVD-ROM discs, recording data to, 610–613
DVD-R/RW discs, copying files to, 129
DVI viewers, availability of, 244
Dynamic Host Configuration Protocol (DHCP), dynamic IPv4 addresses for, 754–755
dynamic IPv4 addresses, using with DHCP, 754–755
dynamic virtual hosting, using with Apache, 459–462

E

e2fsck command, issuing, 594
e2label, using with file systems, 600
echo command
using in TCSH shell, 108
using with shell variables, 68
editors
Emacs, 249–250
Gedit, 248
Gvim and Vim, 250–254
Kate, KEdit, and KJots, 248–249
eject command, using with CD-ROMs, 604
Ekiga VoIP application, features of, 304–305
ELinks line-mode browser, features of, 286
else condition control structure, function of, 79
else keyword, using in TCSH, 85
elsels script, 80, 85
Emacs editor
features of, 249–250
using keyboards with, 251
Emacs mail clients, availability of, 271
enable and disable CUPS command line tools, features of, 512

encrypting file systems, 326
encryption. *See also* GnuPG (GNU Privacy Guard)
    combining with digital signatures, 315
    key used in, 314
    in OpenSSH, 360–361
encryption security payload (ESP), using, 351
endif keyword, using in TCSH shell, 82
endsw keyword, using in TCSH shell, 82–83
env command, listing shell variables with, 96
environment, function of, 8
environment variables
    in Bourne, BASH, and Korn shells, 75–76
    capabilities of, 75
    defining in shells, 94
    in TCSH and C shells, 76
Epiphany GNOME web browser, features of, 286
erasing characters, 21
error messages, placement in standard error, 51–52
esac keyword, effect of, 78–80
ESP (encryption security payload), using, 351
/etc directory, contents of, 537
/etc/exports file
    changing, 765
    entries in, 762–766
    example of, 765
/etc/fstab file
    impact of HAL on, 652–653
    mounting NFS directories by entries in, 768–769
/etc/gdm directory, contents of, 166
/etc/group system file, contents of, 559–560
/etc/gshadow file
    function of, 554
    contents of, 559–560
/etc/hosts file, contents of, 724
/etc/hosts.allow file, function of, 767
/etc/hosts.deny file, function of, 767
/etc/init.d directory, contents of, 402–403
/etc/mdadm.conf file, contents of, 631
/etc/nssswitch.conf file, example of, 729
/etc/profile script, features of, 104–105, 234
/etc/protocols file, contents of, 725
/etc/resolv.conf file, contents of, 725
/etc/services file, contents of, 725
/etc/shadow file, function of, 554
/etc/X11/xorg.conf file, sections of, 150
eth0 device, using ! operator with, 383
Ettercap sniffer program, features of, 739
events
    editing, 43
    editing in C shell history, 61
    performing global substitution on, 61
    re-executing in C shell, 59
    referencing with history utility, 40–42
    referencing in Z shell, 63
    referencing words in, 60
    substituting in C shell history, 59–60
Evolution mail client, using on GNOME desktop, 267–268
execute (x) permission
    explanation of, 562
    invoking, 566
    using binary masks with, 567–568
Exim MTA (mail transfer agent), features of, 478
EXINIT variable, function of, 96
exit command, issuing, 21, 525
EXPN option, disabling in Sendmail, 498
export command
    creating environment variables with, 75, 94
    using with parameter variables, 101
ext3, journaling with, 595
Extended Internet Services Daemon (xinetd). *See* xinetd (Extended Internet Services Daemon)
extensions
    selecting files with, 45
    using with filenames, 116

F

facilities in syslogd daemon, descriptions of, 539–540
FancyIndexing directive, using with Apache web server, 455
fc command, using with history events, 42–43
.fc files, using in SELinux, 340–341
fc-list command, listing fonts with, 26
fd partition type, using with RAID devices, 629
fdi files, using with devices, 654
fdisk tool
    versus df command, 593
    using with file systems, 606–608
FEATURE macro, using with Sendmail, 489–490
Fedora Linux, URL for, 5
Fetchmail, using, 274–275
fg command, execution of, 53–55

FHS (Filesystem Hierarchy Standard). *See also* file systems
and CD-ROM devices, 592
device files in, 590
and floppy disk devices, 592
and hard disk devices, 592
/home directory in, 588
/media directory in, 587
/mnt directory in, 587
and /proc file system in, 589
program directories in, 586
root directory (/) in, 584–585
and /sys, 589–590
system directories in, 585–587
/usr directory in, 587
/var directory in, 588
file command, issuing, 117
file contents, displaying on screen, 119
file extensions, selecting files with, 45
file groups, changing, 565
file owners, changing, 565
File Roller application
archiving and compressing files with, 133–134
starting, 139
file sharing, setting up, 31
file structure, function of, 8
file system labels, using, 600
file system tools
fdisk, 606–608
mkfs, 609–610
mkswap, 610
parted, 608–609
file systems. *See also* distributed network file systems; FHS (Filesystem Hierarchy Standard)
accessing with GNOME Volume Manager, 177
boot and disk check of, 598
checking consistency of, 594
creating for RAID devices, 633–634
encrypting, 326
exporting manually, 765–766
mounting, 583, 593
mounting automatically, 596–601
mounting manually, 601–606
obtaining information about, 593–594
organization of, 583
proc file system, 641
repairing, 594
restoring, 703
sysfs file system, 639–641
File Transfer Protocol (FTP), function of, 423
file types, examples of, 117
FileInfo directive, using with Apache web server, 453
filename completion, performing in TCSH shell, 62
filename expansion, special characters for, 49
filenames
distinguishing from directory names, 123
expanding, 43–47
matching characters in, 44
printing, 50–51
using extensions with, 116
files. *See also* archives; compressed archives
archiving, 134–138
archiving and compressing, 133–134
assigning multiple names to, 130–132
backing up, 81
backing up and restoring, 693–694
backing up to devices, 137
byte-stream format of, 117
compressing with bzip2 utility, 140
compressing with gzip utility, 138–140
compressing with Zip utility, 140–141
copying, 126–129
displaying, 120
displaying detailed information about, 116
erasing, 45, 130
forcing overwriting of, 44
listing, 119
moving, 129
naming, 116
ownership of, 563–565
permission operations for, 564
printing, 121
recovering, 701–703
renaming, 129
safeguarding from redirected output, 109
structure of, 117–119
testing, 78
uncompressing with Zip utility, 141
Filesystem Hierarchy Standard (FHS). *See* FHS (Filesystem Hierarchy Standard)
filters, commands as, 51
find command
locating directories with, 126
searching directories with, 124–125

using -name option with, 124
using -print option with, 124
using -type option with, 126
finger command, obtaining user information with, 301–303
Firefox, enabling JRE (Java Runtime Environment) for, 289
Firefox FTP client
description of, 291
features of, 291–292
Firefox web browser, features of, 284–285
firewall access, blocking in myfilter script, 391
firewall chains, types of, 376
firewalls
configuring, 387–389
function of, 373
and IPtables, 374–375
fixfiles command, using in SELinux, 344–345
Flagship database, features of and website for, 246
Flask architecture, organization of, 327–328
floppy disks
accessing for MS-DOS, 132–133
accessing from KDE desktop, 201, 208
archiving to, 137
mounting, 604
Fluendo, website for, 258
fonts
accessing, 26
adding and removing, 25–26
availability in X Window System, 158
configuring, 26
designating in X Window System, 151–152
installing for Windows, 32
listing, 26
managing, 25
setting in X Window System, 156
for loop control structure, function of, 80
foreach control structure
effect of, 81
using in TCSH shell, 86–88
FormatLog directive, using with Apache web server, 457
FORWARD chain
defining user chain from, 392–393
evaluation of, 380
specifying DROP policy for, 396
use of, 376–377
free command, effect of, 544
Frysk monitoring tool, features of, 544
fsck command, issuing, 594
F-Spot Photo Manager, features of, 256
fstab file
example of, 599–600
integer values in, 598
modifying, 596, 598
FTP (File Transfer Protocol), function of, 423
FTP clients
curl Internet client, 293
features of, 290
Firefox, 291–292
gFTP, 292–293
K Desktop file manager, 292
Konqueror, 292
lftp program, 298
Nautilus, 292
NcFTP program, 299
and network file transfer, 290–291
wget tool, 293
ftp group, using with anonymous FTP, 425
ftp program
automatic login capability of, 297–298
description of, 291
features of, 293–297
macro support in, 297–298
FTP servers
availability of, 424
components of, 423
running Tux as, 444
FTP sites
accessing with rsync, 427–428
components of, 425
performing anonymous login for, 297
performing document searches of, 520
searching files on, 283
FTP user account, requirement of, 425
FTP users
access available to, 424
creating, 426
.ftpaccess files, using with ProFTPD, 436–438
functions script, execution of, 412–413
fuser command, issuing, 603
fuse-smb tool, features of, 31

G

games, downloading, 11
gateway addresses, function of, 719
gateways, encrypting with IPsec, 356–357
GConf configuration editor
features of, 194–196
keys in, 196

gconfig tool, using, 678–679
GDM (GNOME Display Manager)
availability of, 160
changing login screen in, 165
features of, 164–166
function of, 19–20
GDM configuration files, location of, 165
gdmsetup command, issuing, 165
Gedit editor, features of, 248
Gentoo Linux, URL for, 5
-geometry argument, using in X Window System, 155
get command, using with ftp program, 294, 296
getatsc command, issuing, 56
getty program, using, 661
GFS (Global File System), features of, 779–784
gFTP client
description of, 291
features of, 292–293
.gif web file type, description of, 283
GIMP (GNU Image Manipulation Program), features of, 257
gkc (GTK kernel configuration) tool, using, 678
GKrellM system monitors, features of, 545–546
Global File System (GFS), features of, 779–784
global versus exported environmental variables, 76
GNOME (GNU Network Object Model Environment)
About Me preferences in, 26
capabilities of, 9
choosing themes in, 193
configuring, 193
configuring CUPS on, 505
configuring fonts in, 26
configuring networks on, 710
configuring sessions in, 27
Ekiga VoIP application in, 304–305
features of, 169–170
Help browser in, 173–174
include directories for, 193
permissions on, 562–563
quitting, 173
selecting personal photographs in, 26
services-admin tool in, 410
starting programs in, 173
Themes Preferences tool in, 24–25
using switcher with, 20
GNOME applets. *See also* applets
adding, 191
features of, 23–24, 191–192
Window List, 192
Workspace Switcher, 192
GNOME applications
configuring, 195
downloading, 11
using GTK+ widget with, 171
GNOME binaries, location of, 193
GNOME desktop
advisory about removing icons from, 174
applications on, 175
copying files to, 174
creating and editing MIME types on, 267
creating links on, 174–175
displaying contents of tar archives with, 134
displaying with different themes, 172
dragging and dropping files to, 174–175
Evolution mail client in, 267–268
mail clients for, 269–270
menus in, 22–23
moving and copying files in, 172
obtaining documentation for, 14
Search tool in, 125
using with window managers, 175–176
GNOME DESKTOP directory, contents of, 194
GNOME desktop menu, displaying, 175
GNOME developers website, URL for, 13
GNOME Display Manager (GDM). *See* GDM (GNOME Display Manager)
GNOME file manager. *See* Nautilus
GNOME Help Browser, starting, 28
GNOME home directory, contents of, 194
GNOME interface
components of, 171–173
panels in, 172
GNOME Keyring Manager, website for, 317
GNOME libraries, location of, 193
GNOME Network Manager. *See* Network Manager
GNOME Office
applications in, 242–243
website for, 12, 238
GNOME panel objects
adding, 189
Launcher object, 191
Lock object, 191
Logout object, 191
moving, removing, and locking, 189
using application launchers with, 189–190

GNOME panels
adding drawers to, 190–191
adding folders and file launchers to, 190
adding folders to, 191
adding menus to, 191
adding panels to, 187
changing background colors of, 188
configuring, 187–188
displaying, 187
expanded versus unexpanded panels, 188
features of, 187
moving and hiding, 188
GNOME Power Manager, features of, 545
GNOME System Monitor, features of, 543
GNOME Volume Manager
accessing DVD/CD-ROM drives with, 177
features of, 176–178
GNOME web browsers, availability of, 286
gnome-luks-format tool, encrypting file systems with, 326
gnome-nettool utility, features of, 301
GNU archive, URL for, 11
GNU General Public License, significance of, 9
GNU Image Manipulation Program (GIMP), features of, 257
GNU info pages, accessing, 29
GNU Java Compiler, website for, 288
GNU Network Object Model Environment (GNOME). *See* GNOME (GNU Network Object Model Environment)
GNU software, Linux as, 10
GNU SQL database, features of and website for, 12, 246–247
gnubiff mail notification tool, features of, 273
GnuCash finance application, availability of, 244
GnuPG (GNU Privacy Guard). *See also* encryption
checking fingerprints in, 320
decrypting digital signatures in, 322
decrypting messages in, 321–322
encrypting messages in, 321
features of, 316–317
generating private and public keys for, 318
making public keys available in, 318–319
obtaining public keys in, 319–320
protecting keys in, 318
signing messages in, 322
using, 321
validating keys in, 320
gpg command, issuing, 321, 324
gPhoto project, website for, 257
Grand Unified Bootloader (GRUB). *See* GRUB (Grand Unified Bootloader)
graphical user interface (GUI)
forcing exit from, 19
starting from command line, 24
graphics tools, photo management, 256
greater than (>) character
function of, 68
redirecting standard output with, 48–50
as shell symbol, 44
greetarg script, 72
grep command
using with kernel installation, 673
using with list of processes, 405
using with ps command, 543
greylisting, using in Postfix MTA (mail transfer agent), 482–483
group directories, creating, 560–561
groupadd command, issuing, 561
groupdel command, issuing, 561
groupmod command, issuing, 561
groups
changing ownership of, 565
managing, 559–561
growisofs DVD+RW tool, features of, 613
GRUB (Grand Unified Bootloader)
configuring for kernels, 684
features of, 547–550
reinstalling, 18
using with kernel packages, 675–676
gshadow password file, function of, 554
gStreamer
configuring, 260
downloading modules and plug-ins for, 11, 259
installing MP3 support for, 260
plug-ins for, 260
.gtckrc file, description of, 194
gThumb thumbnail image viewer, features of, 257
GTK themes, selecting, 172
GTK+ widget, using with GNOME applications, 171
gtKam graphic tool, downloading, 257
GUI (graphical user interface)
forcing exit from, 19
starting from command line, 24

GUI management tools, availability of, 551–552
GUI sessions, managing with GDM, 164–166
GView graphic tool, downloading, 257
gvim editor, features of, 253
.gz extension, meaning of, 137–138, 220
gzip utility
    compressing files with, 138–140
    using -d option with, 139

H

HAL (Hardware Abstraction Layer)
    and fstab, 596
    function of, 652–653
    implementation of, 653
    relationship to printer device files, 504
    and udev (user devices), 590–592
HAL callouts, function of, 657
HAL links, checking for CD/DVD devices, 261
hald daemon, function of, 653
halt command, issuing, 21
halt file, contents of, 401
hard disk partitions, creating for RAID devices, 631
hard disks formatted for MS-DOS, accessing, 132–133
hard drive partitions
    creating, 637
    creating with LVM, 623–624
    mounting, 605–606
hard links, function of, 130–132
Hardware Abstraction Layer (HAL), relationship to printer device files, 504
hardware virtualization, using with KVM, 687–688
HDTV reception, support for, 263–264
Help resources, accessing, 28–30
history, editing in TCSH shell, 62–63
history command (!)
    editing in C shell, 61
    referencing events with, 41–42, 44, 59–60
    using, 40–41
    using in Z shell, 63
history events
    configuring, 43
    editing, 42–43
    referencing in BASH shell, 40–42
    referencing in Z shell, 63
    tracking, 109
history lists
    displaying commands with, 40–43
    using meta keys with, 41
history number, prompt code for, 100
history utility. *See* C shell history utility
history variable, using in TCSH shell, 111
HISTSIZE variable, default setting for, 43
home directory
    in FHS, contents of, 588
    listing configuration files in, 90
    location of, 118
    name of, 118
    searching, 102
HOME variable, contents of, 96–97
$HOME/* files, descriptions of, 364
host address, form of, 478
host network tool, 301–303
hostnames
    identifying, 723–725
    prompt code for, 100
    representing in BASH shell, 39
    in URLs, 282
hosts
    backing up with Amanda, 695–698
    setting up policies for, 353
hosts.conf file, contents of, 726–727
hotpluggable devices, explanation of, 641
.htaccess file
    contents of, 449
    using for directory-level configuration of Apache, 452–453
ht://Dig search and indexing server, features of, 520
.html web file type, description of, 283
HTTP (Hypertext Transfer Protocol), role in URLs, 282
http_access options, using in Squid, 472
httpd process, checking for, 405
httpd service script, example of, 412, 414–415
hypertext database, components of, 281
Hypertext Transfer Protocol (HTTP), role in URLs, 282

I

IBM database, features of and website for, 246–247
IBM DB2 database software, URL for, 12
ICMP (Internet Control Message Protocol) packets, enabling, 381
ICMP packets, controlling in myfilter script, 391–392

ICPs (Internet Cache Protocols), using with Squid, 473–474
ICQ, features of, 305
id command, determining security context with, 330
identities, function in SELinux, 329–330
if condition control structure
effect of, 78–81
test expressions used with, 82
ifconfig command, issuing, 729–731
if-endif control structure, using in TCSH, 84
ifls script, 84–85
if-then command, effect of, 78–80
if-then structure, using in TCSH, 84–85
ignoreeof feature
description of, 92
using with TCSH shell, 108
IKE, configuring IPsec with, 354–355
ImageMagick program, features of, 257
IMAP (Internet Mail Access Protocol), function of, 498–499
IMAP servers, availability of, 499–500
InfiniBand, support for, 743
info command, issuing, 29
Info pages, accessing, 29–30
information sites, websites for, 14
Informix database, features of and website for, 246–247
init script, functions for, 413
initab, runlevels in, 533
init.d, files in, 402
init.d service scripts, using, 403
initialization files, defining shell parameter variables with, 96
initialization files in TCSH
.login, 112
.logout, 113–114
.tcshrc, 112–113
initialization files, location of, 396
inittab files, relationship to termcap file, 661
Inkscape vector graphics program, features of, 257
INN (InterNetNews) news server
configuration files for, 513–514
features of, 513
implementing, 515
inn.conf file in, 513–514
newsreader accessing, 514
overview support in, 514–515
storage formats for, 514
INPUT chain
defining user chain from, 392–393
evaluation of, 380
in myfilter script, 394
use of, 376–377
input devices
configuration files for, 160
detection of, 662
.inputrc file, creating, 37
INSERT INTO command, issuing, 517
insmod command, loading modules with, 667
installation, preparing for, 18
instances attribute, using in xinetd services, 417
integers, comparing, 77–78
integrity checks, performing with digital signatures, 314
Internet Cache Protocols (ICPs), using with Squid, 473–474
Internet Control Message Protocol (ICMP) packets, enabling, 381
Internet Mail Access Protocol (IMAP), function of, 498–499
Internet Relay Chat (IRC), features of, 305
Internet servers, availability of, 12–13
Internet Software Consortium, URL for, 12
intrusion detection, implementing, 325
iostat command, effect of, 544
IP addresses
advertising with radvd, 749
determining for remote sites, 303
and dynamic virtual hosting, 461–462
obtaining, 717–718
IP aliasing, setting up, 742–743
IP Chains package, features of, 374
IP forwarding, turning on, 396
IP masquerading
implementing, 395
implementing in myfilter script, 391
implementing with netfiltering, 355
and NAT rules, 396
of selected hosts, 396–397
in Sendmail, 491–493
IP spoofing, protecting against, 390
IP table rules, management of, 375
IP Tables firewall, URL for, 12
ip6tables package, features of, 374–375
IP-based virtual hosting, using with Apache web server, 459, 461
IPsec
configuring with racoon tool, 354–355
enabling in kernel, 351

IPsec (*Continued*)
encrypting gateways with, 356–357
modes of, 350
protocols used in, 349–350
security databases, 350
setting up secure two-connections for, 353
tools in, 351
tunnel mode for, 356–357
using certificates with, 355
IPsec connections, configuring, 351–353
IPsec packets, stopping, 355
IPsec transmissions, receiving, 352–353
IPtables
controlling port access with, 382
implementing IP masquerading with, 395
and IP masquerading, 396
options for, 379
and user-defined chains, 380–381
iptables command
adding and modifying chain rules with, 376–378
chain names in, 375
executing, 374
functions of, 377
managing IP table rules with, 375
IPtables modules, location in Netfilter, 375
IPtables netfiltering, effect of, 355
IPtables package, extensibility of, 387
IPv4, dynamic IPv4 addresses for DHCP, 754
IPv4 CIDR addressing, function of, 715–717
IPv4 network addresses, parts of, 712
IPv4 reserved addresses, features of, 717–718
IPv6, advantages of, 711
IPv6 addresses
format of, 720–721
interface identifiers for, 721
types of, 721–723
IPv6 and IPv4 coexistence methods, categories of, 723
IPv6 CIDR addressing, function of, 717
IPv6 router, Linux as, 749
IPv6 stateful autoconfiguration, implementing, 748–749
IPv6 stateless autoconfiguration
capabilities of, 745–746
generating full address for, 746
generating local address for, 746
and router renumbering, 746–747
IRC (Internet Relay Chat), features of, 305
ISO distribution images, downloading with BitTorrent, 220–221
iwconfig command, issuing, 735
iwlist tool, features of, 736
iwpriv command, issuing, 735
iwspy tool, features of, 736

J

Jakarta Project, website for, 288
Java, obtaining Linux ports of, 287
Java 2 Software Development Kit (SDK)
downloading, 287
tools in, 289–290
Java applications
availability of, 288
downloading, 289
Java Runtime Environment (JRE)
enabling for Mozilla and Firefox, 289
installing, 289
job numbers, obtaining, 54
jobs
bringing to foreground, 54–55
canceling, 55–56
canceling and interrupting, 53
referencing, 53–54
running in background, 53–54
suspending and stopping, 55
using notify command with, 54
jobs command, issuing, 53–54
journaling
with ext3, 595
support for, 583, 594–595
JPackage Project, website for, 287
JPEG images, encoding and retrieving data in, 323
.jpeg web file type, description of, 283
J-Pilot, features of, 245
JRE (Java Runtime Environment)
enabling for Mozilla and Firefox, 289
installing, 289

K

K Desktop Environment (KDE). *See* KDE (K Desktop Environment)
K Desktop file manager, FTP capability of, 292
K Desktop file manager window, using as web browser, 285–286
.k5login file, accessing accounts with, 308
kadmin tool, using, 372
Kate editor, features of, 248–249

KDC (key distribution center), role in Kerberos, 369–370
KDE (K Desktop Environment)
accessing system resources in, 202–203
capabilities of, 9
changing settings in, 201
changing virtual desktops for, 205
configuration and administration access with, 199
configuring, 203
configuring CUPS on, 505
configuring fonts in, 26
configuring mouse with, 23
configuring networks on, 710
configuring sessions in, 27
contents of share directory in, 216
copying files with, 23
creating directories in, 202
creating and editing MIME types in, 267
creating URL desktop files in, 203–204
directories and files in, 216
displaying contents of tar archives with, 134
displaying selected directories in, 208
displaying storage media in, 202
features of, 22, 197–198
keyboard shortcuts configuration in, 23
launching applications from, 203
Link To Application entry in, 203–204
mounting devices from, 208
moving windows with, 23
permissions on, 563
and Qt Library, 197–198
quitting, 201
registering file types with, 215
starting applications in, 208
starting up, 19–20
switching users in, 20
Theme manager in, 25
using file manager in, 202–203
using MIME type with, 215
KDE applets, features of, 24
KDE applications
accessing, 201
associating documents with, 207–208
creating desktop files for, 207
setting permissions for, 207
starting, 207
KDE components, locating configuration files for, 215
KDE Control Center, configuring KDE with, 214–216
KDE desktop operations
deleting files, 201
displaying home directory on panel, 201
managing CD-ROMs, DVDs, and floppies, 201
KDE desktop pager, features of, 205
KDE developers library, URL for, 13
.kde directory, contents of, 215
KDE Display Manager (KDM). *See* KDM (KDE Display Manager)
KDE file manager
configuring, 213–214
configuring Navigation panel in, 210–211
copying, moving, deleting and renaming files in, 212
.directory files in, 212
displaying Properties windows for files in, 212
extracting tar archives with, 210
features of, 208–209
navigating directories in, 211–212
opening files from, 210
performing link operations in, 212
searching files in, 211
web and FTP access in, 213
KDE file manager window, components of, 209–210
KDE functions, accessing, 205–206
KDE Help Browser, starting, 28
KDE Help Center, features of, 206–207
KDE interface, confining behavior of, 214
KDE menus, options on, 200–201
KDE panel
adding applications to, 206
applets and panel extensions for, 206
buttons displayed in, 202
configuring position and behavior of, 206
location of, 205
KDE software repository, URL for, 11
KDE Task Manager and Performance Monitor (KSysguard), features of, 546
KDE windows, using, 204–205
kdestroy command, using with Kerberos tickets, 371
kdetv, website for, 262
KDM (KDE Display Manager)
availability of, 160
features of, 166–167
function of, 19–20
KEdit editor, features of, 249–250

KeepAlive directive, using with Apache web server, 449
KeepAliveRequests directive, using with Apache web server, 449–450
Kerberized services, setting up for use of, 371
Kerberos
  authentication process in, 369–371
  description of, 368–369
  website for, 369
Kerberos network authentication protocol, URL for, 12
Kerberos password, managing, 370
Kerberos servers
  configuring, 371–372
  function of, 369
Kerberos tickets, listing, 369
kernel configuration, features of, 679–681
kernel header files, compiling modules with, 670
kernel image, installing manually, 682–683
kernel interface file systems, entries for, 601
Kernel Module Loader (Kmod), function of, 664–665
kernel modules
  location of, 665
  tools for, 664–665
kernel packages
  installing, 674
  using GRUB boot loader, 675–676
kernel RPM packages, downloading, 673
kernel sources, downloading and installing, 677
Kernel-based Virtualization Machine (KVM), features of, 687–688
kernels
  advisory about modification of, 675–676
  compiling and installing, 681–683
  compiling from source code, 676–679
  configuration tools for, 677–679
  configuring, 677
  configuring GRUB for, 684
  definition of, 8, 671
  determining version of, 672
  distribution of, 4
  enabling IPsec in, 351
  installing modules from, 670
  installing new versions of, 673–674
  location of, 674
  major and minor number for, 671
  placing on boot disks, 683
  references for, 672
  retaining copies of, 675
  tunable runtime parameters for, 673
  URL for, 5
  versions of, 671–672
key bindings, configuration of, 37
key distribution center (KDC), role in Kerberos, 369–370
keyboard, configuring with KDE, 23
keyboard shortcuts. *See also* ALT key; CTRL key
  character navigation and editing, 37
  character transposition, 37
  command line interface, 19, 160, 524
  command movement and editing, 38
  configuring in KDE, 23
  erasing command lines, 28
  erasing words, 57
  exit command, 525
  history event editing, 42
  interrupting commands, 36
  job suspension, 55
  KDE desktop navigation, 205
  KDE main menu, 200
  Mail program, 94
  rebooting, 22
  Z shell, 63
keys
  determining tasks associated with, 37
  loading for OpenSSH, 365
  in OpenSSH, 360–361
keyserver, accessing in GnuPG, 319
Kicker. *See* KDE panel
kill command
  canceling jobs with, 55
  issuing, 53
  referencing system process numbers in, 56
killall command, using with xinetd, 407
killproc function, effect of, 412
kinit command, issuing, 370–371
KJots editor, features of, 248–249
klist command, issuing, 369
KMail mail client, features of, 270
Kmod (Kernel Module Loader), function of, 664–665
Knemo network device monitor, configuring, 200
Knoppix Linux, URL for, 5
KOffice applications, availability of, 241–243
KOffice software, URL for, 12
KOffice Suite for KDE, website for, 238

Konqueror. *See* KDE file manager
Konqueror FTP client, description of, 291
Korn shell
    configuration files for, 91
    features of, 89
    website for, 36
KParts, implementation of, 242–243
kpasswd command, issuing, 370
krb5.conf file, contents of, 371
KSysguard (KDE Task Manager and Performance Monitor), features of, 546
KUser tool, features of, 552
KView image viewer, features of, 257
KVM (Kernel-based Virtualization Machine), features of, 687–688

L

-l option, using with commands, 35
LAME, evolution of, 261
LDAP (Lightweight Directory Access Protocol)
    enabling on Thunderbird, 580
    features of, 573
    and NSS (Name Service Switch) service, 580
    and PAM modules, 580
    searching, 579
    support in Sendmail, 485, 487
LDAP configuration files, location of, 574
LDAP directories, implementation of, 573
LDAP directory database
    distinguishing names in, 576–577
    LDIF entries in, 577–578
    schema attributes and classes in, 575–576
LDAP Interchange Format (LDIF) file
    adding records to, 578–579
    contents of, 577–578
LDAP server, configuring, 574–575
LDAP tools, availability of, 579
LDIF (LDAP Interchange Format) file
    adding records to, 578–579
    contents of, 577–578
LDP (Linux Documentation Project), documents available from, 13–14
Leafnode news server, features of, 515
less command
    issuing, 120–121
    using with RPM queries, 225
less than (<) character
    function of, 68
    as shell symbol, 44
Lesser General Public License (LGPL), significance of, 9
lftp program, description of, 291
lftp program, features of, 298
LGPL (Lesser General Public License), significance of, 9
/lib/modules directory, contents of, 665
libraries
    availability of, 233
    shared versus static libraries, 232–233
licenses, protection of open source software by, 9–10
lighthttpd, description of, 444
Lightweight Directory Access Protocol (LDAP). *See* LDAP (Lightweight Directory Access Protocol)
Limit directive, using with Apache web server, 453
line numbers, outputting file contents with, 72
linear level of RAID, function of, 627
linix-wlan project, purpose of, 737
links
    hard links, 131–132
    symbolic links, 131
Linux
    accessing from command line interface, 20–22
    accessing shares on, 31
    components of, 3
    database software available to, 11
    distributions of, 4–5
    history of, 7–8
    office suites available to, 11
    overview of, 8–9
    shutting down from command line, 21–22
    starting from command line, 167–168
    versus Unix, 4
Linux compilers and tools (gcc), URL for, 13
Linux Documentation Project (LDP), documents available from, 13–14
Linux Foundation website, accessing, 8
Linux Game Tome, URL for, 11
Linux sessions, ending, 21
Linux software. *See also* software
    applying RAID levels to, 627–628
    availability of, 10–13
    downloading, 11

Linux systems
accessing, 19–22
setting up file sharing on, 31
shutting down and restarting, 19–20
LinuxTV.org, links at, 262
Listen directive, using with Apache web server, 450, 459
ln command
issuing, 130–132
using -s option with, 131
LoadModule directive, using with Apache web server, 450
log files
checking for suspicious activity, 325
customizing in Apache, 457
generating and managing in Apache, 458
LogFormat directive, using with Apache web server, 457
logging in and out, 20–21
logical operations
performing with test command, 77–78
in TCSH shell, 83
Logical Volume Management (LVM). *See* LVM (Logical Volume Management)
.login initialization file in TCSH
description of, 111
function of, 112
login shell, configuring, 101
logout command
issuing, 21, 94
using in TCSH shell, 108
.logout initialization file in TCSH
description of, 111
function of, 113–114
logresolve utility, using with Apache log files, 458
logs
using for Squid access, 474
viewing, 537
loop control structures, types of, 77
loopback interface, relationship to mail servers, 478
lpadmin CUPS command line tool, features of, 506, 511
lpinfo CUPS command line tool, features of, 512
lpnum executable file, 71–72
lpoptions CUPS command line tool, features of, 511–512
lpq and lpstat print clients, using with CUPS, 510
lpq command, issuing, 121
lpr command
issuing, 119, 121
using, 51, 53
lpr print client, using with CUPS, 509
lprm command, issuing, 121
lprm print client, using with CUPS, 510
ls command
issuing, 37, 119, 121, 123
renaming, 92
ls -l command, issuing, 116
lsof command, issuing, 603
Luks, encrypting file systems with, 326
lvcreate command, issuing, 620
LVM (Logical Volume Management)
device-mapper used by, 621–622
distribution configuration tools for, 617
example of, 623–624
function of, 615
versus RAID, 616
replacing drives with, 622–623
structure of, 616–617
LVM devices, names for, 621–622
LVM groups, managing, 619–620
LVM information, displaying, 619
LVM logical volumes, managing, 620–621
LVM physical volumes, managing, 619
LVM snapshots, creating, 625
LVM tools, using commands as, 617–622
LVM volume groups, activating, 620
LVMs, creating during installation, 617
Lynx line-mode browser, features of, 286

M

mail, accessing on remote POP mail servers, 274–275
mail address, components of, 478
mail clients
command line clients, 271–273
Emacs, 271
Evolution, 267–268
in GNOME, 269–270
KMail, 270
MIME (Multipurpose Internet Mail Extensions), 266
notifications of received mail, 273–274
protocols for, 267
signature files for, 265
SquirrelMail, 270–271
Thunderbird, 268–269

mail delivery agents (MDAs), features of, 477
mail exchange (MX) records, relationship to mail servers, 478–479
Mail Notification tool, features of, 273
Mail program, keyboard shortcut for, 94
mail servers. *See also* Sendmail mail server
    IMAP (Internet Mail Access Protocol), 498–499
    POP (Post Office Protocol), 498–499
    Postfix MTA (mail transfer agent), 479–484, 501
    Sendmail mail server, 487
mail transfer agents (MTAs). *See* MTAs (mail transfer agents)
mail user agents (MUAs), definition of, 477
Mail utility command line client, features of, 272
mail variable, using in TCSH shell, 111
MAILER macro, using with Sendmail, 488, 490
mailer table, using with Sendmail, 495
mailing lists, subscribing to, 275
Mailman, managing mailing lists with, 275
.mailrc initialization file, 273
Majordomo, website for, 275
make command, using with kernels, 681–682
make policy command, using in SELinux, 344
MAKEDEV command, creating device files with, 657–658
Makefiles
    editing, 233
    using with configuration scripts, 231
man command, issuing, 29
Man pages, accessing, 29
mangle table
    identifying, 386
    relationship to packet filtering, 375
    targets for, 386
manual devices, creating, 658–659
MASQ gate, explanation of, 395
masquerading. *See* IP masquerading
MaxClients directive, using with Apache web server, 451
MaxDB database, features of and website for, 246–247
MaxRequestsPerChild directive, using with Apache web server, 451
MCS (Multi-Category Security), extending SELinux with, 331–332, 336
MD driver, using with RAID devices, 629
mdadm
    -create options for, 633
    performing RAID administration with, 629
mdadm.conf, options for, 631
mdadmd daemon, using with Multipath, 628
MDAs (mail delivery agents), features of, 477
mdir command, using with MS-DOS disks, 133
/media directory in FHS, contents of, 587
media players, access to, 262–263
menuconfig tool, using with kernels, 677–678
Mepis Linux, URL for, 5
meta keys, using with history lists, 41
metacharacters, quoting, 68–69
mget command, using with ftp program, 296
Microsoft Office, running on Linux, 238–239
MIME (Multipurpose Internet Mail Extensions)
    features of, 266
    standard associations for, 267
MIME protocols, use with mail clients, 267
MIME types
    and Apache web server, 454–455
    creating and editing on GNOME and KDE, 267
mime.types file, entries in, 266–267
mingetty program, using, 661
Minix program, significance of, 7
mirroring level of RAID, function of, 628
mkdir command
    issuing, 122–123
    using with ftp program, 294
mkfs tool, using with file systems, 609–610
mkinitrd command, creating RAM disk image files with, 685
mkisofs command, using with CDs and DVDs, 611–612
mknod command, creating devices with, 659–660
mkswap tool, using with file systems, 610
MLS (Multi-Level Security), extending SELinux with, 331–332, 336
/mnt directory in FHS, contents of, 587
.mod module files, creating in SELinux, 341
mod_ssl implementation of SSL
    configuring, 466
    features of, 465
modems
    installing and managing, 660–661
    symbolic link file for, 646

modinfo command, discovering module parameters with, 666
modprobe command
installing modules with, 666–667
using in connection tracking, 384
using with depmod command, 666
modprobe configuration, actions associated with, 667–668
module dependencies, detecting, 666
module RAM disks, using, 684–685. *See also* RAM (random access memory)
modules
compiling with kernel header files, 670
downloading drivers for installation of, 669
installing from kernel, 670
kernel module tools, 664–665
loading, 667
managing with modprobe, 666
unloading, 667
use of, 664
Mono, .NET support provided by, 613–614
more command, issuing, 51, 120–121
motherboard RAID support, availability of, 626–627
mount command
using with file systems, 596, 602–603
using with NFS (network file systems), 769
mountpoint, definition of, 602
mouse, configuring with KDE, 23
moving files, 129
Mozilla, enabling JRE (Java Runtime Environment) for, 289
Mozilla framework, use of, 283–284
Mozilla project, Thunderbird mail client offered by, 268–269
MP3, alternative to, 261
MP3 gstreamer plugin, downloading, 11
.mpeg web file type, description of, 283
MPlayer, features of, 262–263
MPMs (multiprocessing modules)
configuring for Apache, 450–451
using with Apache, 447
mput command, using with ftp program, 296
MS-DOS disks, accessing, 132–133
MTAs (mail transfer agents)
definition of, 477
features of, 477–478
mtools, using, 132–133
mtr network tool, features of, 302
MUAs (mail user agents), definition of, 477
multicast address, explanation of, 721–723
Multi-Category Security (MCS), extending SELinux with, 331–332, 336
MULTICS (Multiplexed Information and Computing Service), significance of, 6–7
Multi-Level Security (MLS), extending SELinux with, 331–332, 336
multimedia applications, 258
CD burners and rippers, 261
gStreamer, 258–260
sound, 260–261
video, 262–264
multimedia support, installing, 27
Multipath, relationship to RAID levels, 628
Multiplexed Information and Computing Service (MULTICS), significance of, 6–7
multiprocessing modules (MPMs), configuring for Apache, 450–451
Multipurpose Internet Mail Extensions (MIME)
features of, 266
standard associations for, 267
music files, compression of, 261
Mutt command line mail client, features of, 271
Mutt newsgroup, accessing, 271
mv command
issuing, 129
meaning of, 124
using with directories, 129–130
MX (mail exchange) records, relationship to mail servers, 478–479
myarchprog script, 138
myargs script, 73
.my.cnf file, using, 518–519
myfilter IPtables script
allowing local network access in, 391
blocking outside initiated access in, 391
controlling ICMP packets in, 391–392
creating, 387–389
firewall outside access in, 391
implementing masquerading in, 391
LAN configuration in, 394
listing rules in, 392
protecting against IP spoofing in, 390
server access in, 390–391
setting up DROP policy in, 389–390
user-defined rules in, 392–393
mylist script, 86
mylistarg script, 87
MySQL database, features of and website for, 246

MySQL database server
configuring, 517–518
global configuration of, 518
managing with mysql and mysqladmin, 519–520
structure of, 517
user configuration of, 518–519
MySQL database software, URL for, 12
MySQL tools, availability of, 519

N

-name option, using with find command, 124
name server addresses, function of, 719–720
Name Service Switch (NSS). *See* NSS (Name Service Switch)
NAT (network address translation), explanation of, 384
NAT chains, types of, 376
NAT operations, types of, 385
NAT redirection, implementing, 386
NAT rules
adding, 384–385
and IP masquerading, 396
NAT table
chains in, 385
valid targets for, 386
NAT tasks, executing, 374
Nautilus
adding emblems in, 181
Browser view of, 180
displaying files and folders in, 180–181
features of, 178
File menu in, 182
as FTP browser, 186
grouping files in, 183
navigating Spatial and Browser views in, 182
pixmaps directory in, 185
removing files in, 182
renaming files in, 182–183
setting display properties for, 186
setting preferences for, 186
Spatial view of, 180
starting applications and files in, 183–184
using application launchers with, 184
using Open With option with, 183–184
Nautilus FTP client
description of, 291
features of, 292
Nautilus menu, options on, 181
Nautilus properties panels
Basic panel, 184–185
Emblems panel, 185
Notes panel, 186
Open With panel, 185
Permissions panel, 185
Nautilus sidebar
History view on, 180
Information pane on, 180
Notes view on, 180
Places view on, 180
Tree view on, 180
Nautilus windows
Browser view of, 178–179
Spatial view of, 178–179
zooming in and out of views in, 179
NcFTP client, description of, 291
NcFTP program, features of, 299
NCSA web server, description of, 444
.NET support, availability of, 613–614
net_traversal option, adding to racoon, 355
Netfilter rules, using with NFS servers, 768
Netfilter software package
features of, 374
modules of, 375
netfiltering, implementing IP masquerading with, 355
netgroups, setting up with NIS, 774
netmasks, relationship to TCP/IP network addresses, 713–714
.netrc ftp configuration file, using, 297–298
Netscape Enterprise Server, description of, 444
Netscape source code, availability of, 284
netstat program, features of, 742
network address translation (NAT), explanation of, 384
network connections
configuring with iwconfig, 735
monitoring with netstat program, 742
network file systems (NFS). *See* NFS (network file systems)
Network Information System (NIS). *See* NIS (Network Information System)
network interfaces and routes, overview of, 729–733
Network Manager
features of, 733–734
manual wireless configurations in, 735–737

network monitoring tools
Ettercap sniffer program, 739
netstat, 742
ping program, 739
tcpdump, 741
Wireshark, 739–741
network packets, capturing with tcpdump, 741
network security applications, websites for, 374
network tools
finger command, 301–303
gnome-nettool utility, 301
host command, 301–303
Kerberized network tools, 371
ping command, 301–303
traceroute command, 301–304
networks, configuring on GNOME and KDE, 710
newline, prompt code for, 100
newrole command, issuing in SELinux, 330
news servers
features of, 512
INN (InterNetNews) news server, 513–515
Leafnode, 515
specifying, 100–101
news sites, websites for, 14
news transport agents, using with Usenet news, 278–279
newsbin program, installing, 32
newsgroups
availability of, 15
for Mutt, 271
topics for, 276
newsreaders, reading Usenet articles with, 277
NFS (network file systems)
configuring, 762–766
file and directory security of, 766
mounting automatically, 768–770
mounting manually, 769
mounting on demand, 770
and NFS daemons, 762
NFSv4, 761
options for, 764
starting and stopping, 762
user-level access in, 764
NFS host entries, types of, 762, 764
NFS servers, controlling access to, 766–768
NFS4, setting up ACLs with, 766
NIS (Network Information System)
development of, 771
relationship to NFS, 770
setting up netgroups with, 774
NIS clients, configuring, 774–775
NIS databases, creating, 773
NIS domains, defining, 771–772
NIS servers
controlling access to, 773
setting options for, 772
specifying shared files for, 772–773
NIS servers, configuring, 771–773
NNTPSERVER variable
using, 100–101
using with newsreaders, 277–278
noauto option, using with file systems, 601
noclobber feature
description of, 92
turning on, 108
using with TCSH shell, 109
noglob feature
description of, 92
using with TCSH shell, 109
NOT, logical operator for, 78
notification tools, using for mail, 273–274
notify command, using with jobs, 54
NSS (Name Service Switch)
and DNS (Domain Name Service), 727–729
relationship to LDAP, 580
nsswitch.conf, specifying configuration files with, 775
NTFS partitions, mounting, 239
ntfs type, specifying for Windows partitions, 600–601
number sign (#) prompt, meaning of, 21

O

.o extension, meaning of, 665
-o option, using with set command, 92
octal number, calculating for absolute permissions, 567
off option, using with chkconfig, 408
Office, running on Linux, 238–239
office suites, availability of, 11–12
Ogg Vorbis, website for, 261
online resources, availability of, 13
The Open Group (TOG) website, accessing, 146
Open Secure Shell, URL for, 12
open source software, Linux as, 9–10
OpenOffice software
URL for, 12
website for, 238

OpenOffice.org applications, 239
Calc spreadsheet, 240
database access, 241
Draw tool, 240
Math tool, 240
Writer word processor, 240
OpenPGP protocol, use with mail clients, 267
OpenPGP Public Keyserver project, website for, 319
OpenPrinting database, website for, 504
OpenSSH. *See also* SSH (Secure Shell)
authentication in, 361
authorized keys for, 365
encryption in, 360–361
loading keys for, 365
port forwarding in, 367–368
tools for, 361–362
tunneling in, 367–368
website for, 360
OpenSSL, availability of, 465
openSUSE Linux, URL for, 5
operating systems, function of, 6
operators in syslogd daemon, descriptions of, 539–540
options, using in commands, 28
Options directive, using with Apache web server, 453
Options directive, using with SSIs (server-side includes), 462
OR, logical operator for, 78
Oracle database, features of and website for, 246–247
Oracle database software, URL for, 12
OSTYPE macro, using with Sendmail, 490–491
OUTPUT chain
in myfilter script, 394
use of, 376–377, 385
overwriting (>!) shell symbol, execution of, 44
ownership permissions, setting, 569

P

packet filtering
adding and changing rules in, 376–378
and chains, 375–376
executing, 374–375
implementation of, 375
relationship to firewalls, 373
and targets, 376
packet mangling, 386
packets
accepting and denying, 379–380
changing addresses of, 385
detecting tracking information for, 383
enabling ICMP packets, 381
routes of, 731
setting services for, 385
tracking, 384
PAM configuration files, contents of, 581
PAMs (Pluggable Authentication Modules), using with LDAP, 580–582
panels. *See* GNOME panels
Parallel Virtual File System (PVFS), features of, 777–778
parameter variables, exporting, 101
parent directory. *See also* directories
identifying, 125
referencing, 124
setting permissions for, 568
parity level of RAID, function of, 628
parted tool
GUI interface available to, 631
using with file systems, 608–609
partitions
creating for RAID devices, 631
creating on different hard drives, 623–624
passwd command
controlling user passwords with, 556
description of, 525
use by root user, 524
password files
fields in, 553
location of, 553–554
password tools, using, 554
passwords
controlling user passwords, 556
entering and protecting, 21
managing in Kerberos, 370
PATH variables
adding directories to, 102
assignment of, 101
contents of, 98–99, 234
creating assignment entry for, 234
customizing, 234–235
defining, 96
pathmunge function, using with PATH variable, 234
pathnames. *See also* URL pathnames
of directories, 118–119
types of, 119
in URLs, 282
of variables, 110

patterns, generating, 47
PCHDTV video card, availability of, 263
PCMCIA devices, support for, 664
PDA access, availability of, 245
PDF viewers, availability of, 244
percent (%) symbol
    preceding job numbers with, 54
    using with Apache log files, 457–458
performance analysis tools and processes
    Frysk, 544
    GKrellM, 545–546
    GNOME Power Manager, 545
    GNOME System Monitor, 543
    KSysguard, 546
    ps command, 543
    System Tap, 544
    vmstat, top, free, Xload, iostat, and sar, 544
period (.). *See also* dot (.) command
    representing directories with, 68, 90
    using with extensions, 116
Perl, URL for, 13
permission symbols, using, 566
permissions
    absolute method of, 566–568
    categories of, 562
    changing with chmod command, 563
    on GNOME, 562–563
    on KDE, 563
    representing empty permissions, 562
    setting defaults for, 570–571
    setting on directories, 568
    setting ownership permissions, 569
    sticky bit permissions, 569–570
personal information, configuring, 26–27
PGP (Pretty Good Privacy), websites for, 314
PGP secret keys, managing, 317
photo management tools, availability of, 256
PHP (PHP: Hypertext Preprocessor), features of, 463
PID (process ID), significance of, 55–56
pilot-link package, contents of, 245
ping command
    checking system connections with, 301–303
    features of, 739
ping operations, controlling relative to ICMP packets, 381
pipe (|)
    connecting commands with, 51
    as shell symbol, 44
    using with standard output, 49
Places menu in GNOME, capabilities of, 22–23
Pluggable Authentication Modules (PAMs), using with LDAP, 580–582
.png web file type, description of, 283
policies. *See also* security policies
    adding to SPD (policy database), 351
    versus properties for devices, 655
    for removable storage devices, 656
    setting up for receiving hosts, 353
policy database (SPD), adding policies to, 351
POP (Post Office Protocol), function of, 498–499
POP mail servers, accessing mail on, 274–275
POP servers, availability of, 499–500
port access, controlling, 382
port forwarding, implementing in OpenSSH, 367–368
portmapper service, using with NFS servers, 767
ports, specifying for packets, 385
Post Office Protocol (POP), function of, 498–499
Postfix mail server, URL for, 12
Postfix MTA (mail transfer agent)
    commands in, 479–480
    configuring, 480
    configuring for SpamAssassin, 501
    controlling clients, senders, and recipients in, 483–484
    controlling user and host access in, 483–484
    direct connections for, 481
    features of, 479
    greylisting support in, 482–483
    header and body checks in, 483
    on local networks, 481
    main.cf file in, 480
    masquerade_domains parameter for, 482
    setting network parameters for, 480–481
    virtual domains and virtual accounts for, 482
PostgreSQL database, features of and website for, 12, 246, 520
POSTROUTING chain, use of, 376, 385
PostScript viewers, availability of, 244
pound (#) symbol
    as prompt, 21, 36
    using with ServerName directive, 452
    using with service script tags, 413–414
    using with xinetd services, 409–410
PowerDVD, website for, 262

PPP access, implementing from command line, 737–738
Preferences menu in GNOME, capabilities of, 23
Preferred Applications tool, features of, 184
PREROUTING chain, use of, 376, 385
PreSession directory, contents of, 166
-print option, using with find command, 124
printer device files, names for, 504, 643
printer drivers, downloading, 504
printers, installing with CUPS, 505–507
printfile script, 75–76
printing
    filenames, 50–51
    files, 53, 121
printing services, using CUPS for, 503–504
priorities in syslogd daemon, descriptions of, 539–540
private keys, use in OpenSSH, 360–361
/proc file system, function of, 641
/proc file system, relationship to FHS, 589
processes
    ending, 55–56
    listing, 405
    listing with ps command, 543
procmail, configuring to use SpamAssassin, 500–501
profile script, location and contents of, 103
profile scripts
    /etc/login.access, 555–556
    /etc/login.defs, 555
    /etc/skel, 555
    function of, 554–555
ProFTPD (Professional FTP Daemon)
    and anonymous access, 438–440
    authentication in, 436
    directives used with, 437
    and guest login, 439–440
    installing and startup of, 436
    proftpd.conf file in, 436–438
    using .ftpaccess files with, 436–438
    using RequireValidShell directive with, 439
    and virtual FTP servers, 440–442
    website for, 12
program directories, contents of, 537, 586
programs, location of, 233
prompt command, using with ftp program, 296
prompt variables
    creating command line prompts with, 109
    using in TCSH shell, 109–110
prompts
    codes for, 100
    configuring, 99–100
    creating, 109
    using with shells, 36
property entries, using with devices, 654–655
proxies, function of, 374
proxy servers, features of, 467–468
ps command
    canceling jobs with, 55–56
    listing processes with, 543
PS1 and PS2 variables, contents of, 99
pub directory, contents of, 427
public keys
    checking in RPM packages, 324–325
    importing for software packages, 323
    in OpenSSH, 360–361
    validating for software packages, 324
public licenses, protection of open source software by, 9–10
public-key encryption, function of, 314
put command, using with ftp program, 294, 296
PVFS (Parallel Virtual File System), features of, 777–778
pwd command, issuing, 121–122

Q

-q and -qa options, using with RPM packages, 224–225
qconf tool, using, 679
Qmail MTA (mail transfer agent), features of, 478
QPL (Qt Public License), significance of, 10
Qpopper POP server, website for, 500
Qt Library, advantages of, 197–198
Qt Public License (QPL), significance of, 10
.QT web file type, description of, 283
question mark (?)
    function of, 68
    matching single characters with, 45–46
    as shell symbol, 44
quota command, issuing, 572–573
quota tools
    edquota, 571–572
    using, 571
quotacheck command, issuing, 572
quotaoff command, issuing, 572
quotaon command, issuing, 572

quotes
distinguishing types of, 70
using with commands, 69
using with metacharacters, 68–69

R

r (read) permission
executing, 566
explanation of, 562
-r option, using with cp command, 130
racoon tool
configuring gateway connections with, 357
configuring IPsec with, 351, 354–355
radvd, advertising IP addresses with, 749
RAID (Redundant Array of Independent Disks)
configuring, 631–632
configuring bootable RAID, 635
and dmraid, 626–627
example of, 636–637
function of, 615, 625–626
versus LVM, 616
RAID administration, performing with mdadm, 629
RAID arrays
creating, 632
managing, 634
monitoring, 634–635
starting and stopping, 634
RAID devices
booting from, 629
and corresponding hard disk partitions, 635–636
creating and installing, 630
creating file systems for, 633–634
creating hard disk partitions for, 631
creating spare groups for, 633
fd partition type used with, 629
location of configuration files for, 631
MD driver used with, 629
RAID file systems, adding, 630
RAID levels
applying ot Linux software, 627–628
and Multipath, 628
RAM disk image files, creating, 685
RAM (random access memory), listing amount of, 544. *See also* module RAM disks
range declaration, using with dynamic IPv4 addresses for DHCP, 754
rar archives, purchasing archiver for, 133
RBAC (Role Based Access Control) mode, use in SELinux, 328, 336
rc script, function of, 403
rcconf tool
managing services with, 410–412
using with runlevels, 536
rc.local file, contents of, 401
rcp command, entering, 307–309
rc.sysinit file, contents of, 401
RDBMS (relational database management systems)
SQL databases as, 245–247
structure of, 516
read (r) permission
executing, 566
explanation of, 562
Readline, editing capabilities provided by, 37
reboot, forcing, 22
received mail, notifications of, 273–274
receiving hosts, setting up policies for, 353
Red Hat Global File System, features of, 779–784
Red Hat Linux
runlevels for, 405
using virt-install script in, 690–691
website for, 5
Red Hat Package Manager (RPM)
features of, 221–222
packaging software with, 235–236
Red Hat public key, downloading, 323
Red Hat virt-manager, features of, 686–687
Redirect directive, using with Apache web server, 454
REDIRECT target, using, 386
redirected output, safeguarding files from, 109
redirection operations, combining, 50
redirection operator (>>)
as shell symbol, execution of, 44
using with standard output, 48–50, 52
Redundant Array of Independent Disks (RAID). *See* RAID (Redundant Array of Independent Disks)
refpolicy SELinux source file, contents of, 342
reget command, using with ftp program, 295
ReiserFS, journaling with, 595
reject and accept CUPS command line tools, features of, 512
relational database management systems (RDBMS)
SQL databases as, 245–247
structure of, 516

relational operators, using in TCSH shell, 83
relative pathname, explanation of, 119
RELAY rule, using with Sendmail, 497–498
remote access commands, function of, 307–308
remote access information, obtaining, 308
remote access permission, obtaining, 308
remote login, performing, 309
remote POP mail servers, accessing mail on, 274–275
remote printers, configuring on CUPS, 507
remote sites, determining IP addresses of, 303
remote X applications, using, 148–149
removable devices and media, using, 27
removable media, managing with GNOME Volume Manager, 176–178
removable storage devices, policies for, 656
rename command, using with ftp program, 294
repeat structure, using in TCSH shell, 86
repquota command, issuing, 572–573
RequireValidShell directive, using with ProFTPD, 439
Resource Group Manager, using with GFS, 782
Restart option, using, 19–20
restore command, issuing, 701–703
restorecon command, using in SELinux, 345
reverse proxy cache, configuring for Squid, 475
rgmanager, using with GFS, 782
Ritchie, Dennis, 7
rlogin command, issuing, 309
rm command
    erasing files and directories with, 130
    issuing, 45, 123
    using with hard links, 132
rmdir command
    issuing, 122–123
    using with ftp program, 294
rmmod command, unloading modules with, 667
Role Based Access Control (RBAC) method, use in SELinux, 336
Role Based Access Control (RBAC) mode, use in SELinux, 328
root directory (/)
    contents of, 118
    indicating, 120
    role in FHS, 584–585
root user
    logging in as, 524
    use of passwd command by, 524
root user account, logging into, 524
root user-level access, controlling, 525–526
root window in X Window System
    controlling display of, 156
    setting features of, 160
rootnoverify option, using with GRUB, 548
rotatelogs utility, using with Apache log files, 458
route command, issuing, 732–733
RPM (Red Hat Package Manager)
    features of, 221–222
    packaging software with, 235–236
rpm command
    installing and updating packages with, 226
    issuing, 222
    using with kernel installation, 673
RPM database, rebuilding, 227
.rpm extension, meaning of, 10, 220
RPM installation, verifying, 226–227
RPM package repository, URL for, 11
RPM packages
    checking public keys in, 324–325
    displaying information from, 225
    installing and uninstalling, 222
    naming conventions for, 222
    with noarch term, 222
    querying information with, 224–225
    removing, 226
rsh command, entering, 307, 309–310
RSSOwl tool, using with GNOME, 286
rsync command
    accessing FTP sites with, 427–428
    issuing, 694
rsync mirroring, implementing, 429
rsync servers, configuring, 428–429
rules.d directory, contents of, 643–645
runlevel command, issuing, 534
runlevel links, using update-rc.d tool with, 411
runlevels. *See also* states
    availability of, 531–532
    changing with telinit, 149, 162, 533–534
    detecting, 403
    in initab, 533
    for Red Hat and Debian, 405
    toggling services for, 536
ruptime command, issuing, 308
rwho command, issuing, 308

S

-s option, using with ln command, 131
sa (security associations), using, 351–352
SAD (security database), adding security associations to, 351

Samba
browsing Windows systems with, 177
enabling CUPS on, 507
using to access Windows systems, 30–31
website for, 12
SAP database, features of and website for, 246–247
sar command, effect of, 544
SASL (Simple Authentication and Security Layer), using with Sendmail, 496
savhistory variable, using in TCSH shell, 111
scandvb tool, features of, 263–264
scheduled tasks, organizing, 529
scp client, features of, 366–367
scp command, issuing, 309
Scribus desktop publishing tool, website for, 238
script arguments, using, 71–73. *See also* arguments
script files, setting executable permission for, 71
Seahorse Encryption Key Manager, accessing, 170–171
seaudit tool, checking SELinux messages with, 334
Secure Shell (SSH), implementations of, 359. *See also* OpenSSH
Secure Sockets Layer (SSL), using with web servers, 464–466
security
of Sendmail mail server, 496–498
in Squid, 470–473
security associations (sa), using, 351–352
security context
checking in SELinux, 333
determining for security identity, 330
relationship to Flask, 327–328
in SELinux, 331
using, 345
security database (SAD), adding security associations to, 351
security policies, using, 352. *See also* policies
Security-Enhanced Linux (SELinux). *See* SELinux (Security-Enhanced Linux)
seinfo command, using with SELinux, 332–333
SELECT command, issuing, 517
SELinux (Security-Enhanced Linux)
adding users in, 345–346
allow keyword in, 339–340
changing roles in, 329
checking security context in, 333
checking statistics for, 332–333
checking status of, 332–333
checkmodule command in, 341
configuring, 337
contexts/files directory in, 347
default_context file in, 346
default_types file in, 346
description of, 327
disabling, 332
domains in, 330
file_context.homedirs directory in, 347
identities in, 329–330
implementation of policies in, 337
initrc_context file in, 346
labeling in, 331
managing users in, 346
MLS (Multi-Level Security) in, 331–332
policies in, 331
roles in, 330
runtime security contexts and types in, 346–347
security contexts in, 331
security models used in, 328
system administration access in, 328–329
transitions in, 331
turning off, 332
types in, 330
using security contexts in, 345
SELinux management tools
apol Policy Analysis tool, 334
audit2allow, 334–335
chcon, 334
semanage, 334
SELinux messages, checking, 334
SELinux modules, compiling, 341
SELinux policies, creating, 344–345
SELinux policy configuration files
application configuration, 344
changing, 340–341
.fc and .te files, 340–341
interface files, 342
module files, 343
policy module tools, 343–344
roles, 343
security context files, 343
types files, 343
SELinux policy methods
RBAC (Role Based Access Control), 336
type enforcement, 336
SELinux policy rules
access vectors, 339–340

composition of, 337
constraint rules, 340
file contexts, 339
role allow rules, 340
transition and vector rule macros, 340
type and role declarations, 338
user roles, 339
SELinux reference policy, using, 335–336
SELinux source configuration, using, 341–342
SELinux users, permissions retained by, 336
semanage command, applying to users, 346
semanage tool in SELinux, features of, 334
semanage_module command, using, 344
semicolon (;), separating commands with, 37, 44
semodule_package command, using, 341
Sendmail mail server
configuring, 487–491
configuring for centralized mail server, 494–495
configuring for simple network configuration, 494
configuring workstations with direct ISP connections, 495
disabling EXPN option in, 498
disabling VRFY option in, 498
features of, 402, 484–485
as mail server or mail client, 493
masquerading, 491–493
redirect feature in, 489–490
security of, 496–498
support for aliases and LDAP, 485, 487
URL for, 12
using access.db file with, 497
using define command with, 490
using divert command with, 491
using dnl command with, 491
using mailer table with, 495
using RELAY rule with, 497–498
and virtual domains, 496
sendmail.cf file
definitions inf, 488
location of, 487
serial ports, types of, 660
server management tools, availability of, 536
ServerAdmin directive, using with Apache web server, 451
ServerLimit directive, using with Apache web server, 451
ServerName directive, using with Apache web server, 451–452
ServerRoot directive, using with Apache web server, 449
servers, starting with init.d, 403
server-side includes (SSIs). *See* SSIs (server-side includes)
ServerTokens directive, using with Apache web server, 449
service command, issuing, 403, 405
service scripts
example of, 414–415
function of, 412–413
installing, 415
tags for, 413–414
services
accessing with Kerberos, 370
configuring automatic startup of, 402
controlling access to, 374
definition of, 404
displaying startup information for, 409
listing with chkconfig, 407–408
port numbers and labels for, 382
removing and adding with chkconfig, 409
restoring to chkconfig default, 408
setting for packets, 385
starting and stopping, 403
starting and stopping with chkconfig, 408–409
starting and stopping with service scripts, 406
starting automatically, 406–407
starting directly, 405
turning on and off, 410
services-admin tool, features of, 536
sessions, configuring in desktops, 27
sestatus command, using with SELinux, 332
set command
arguments for, 93
defining C shell history with, 58
using with shell variables, 68
using with TCSH shell, 108
setenv command, defining environment variables with, 76
setfiles tool, using in SELinux, 345
setkey tool
configuring connections with, 351–353
using with IPsec, 351
seusers file, root entry in, 347
sftp and sftp-server clients, features of, 367
shadow password support, turning off, 553
share directory in KDE, contents of, 216
shares, accessing on Linux systems, 31

shell commands, entering into script files, 70–71
shell operations, controlling, 93–94
shell parameter variables
features of, 95–97
and initialization files, 96
listing, 96
shell prompt, configuring, 99–100
shell scripts
capabilities of, 65
defining variables in, 73
definition of, 66, 70
executing, 71
using execute permissions with, 567
shell symbols
execution of, 44
matching, 46–47
shell variables
assigning values to, 67–68
availability of, 66
and command arguments, 68
for configuring systems, 109–111
defining and evaluating, 66–67
definition of, 66
shell version, prompt code for, 100
shells. *See also* BASH shell; C shell; TCSH shell; Z shell
accessing, 35
defining environment variables in, 94
definition of, 65
initialization and configuration files for, 90
types of, 35, 89
user shells versus subshells, 73–74
using prompts with, 36
showrgb command, using with X Window System, 159
shutdown commands
description of, 525
issuing, 19–21, 534–535
location of, 401
signature files, using with mail clients, 265
Simple Authentication and Security Layer (SASL), using with Sendmail, 496
single quotes (')
versus back quotes (`), 69
using with commands, 69
slash (/), use with BASH command line, 39
slogin command, issuing, 309
slrn newsreader, features of, 278
smb command, issuing, 31
smb.conf file, contents of, 466
SNAT targets, using, 385, 396
.so extension, meaning of, 233
software. *See also* Linux software
compiling, 231–232
decompressing and extracting, 228–229
decompressing separately, 229–230
extracting, 230–231
installing from compressed archives, 228–233
packaging with RPM, 235–236
query options for, 224–225
software packages
checking digital signatures in, 323–325
installing and updating with rpm, 226
types of, 219–220
software repositories
contents of, 30
updating software from, 10–11
sort command, using, 51
sound applications, support for, 260–261
sound devices, installing, 662–663
SourceForge, website for, 9, 11
SpamAssassin, features of, 500–501
SPD (policy database), adding policies to, 351
spdadd instruction, using with security policies, 352
special characters, disabling in user shell, 109
spool directories, relationship to printing, 505
SQL databases, as RDBMS, 245–247
SQL query language, features of, 516–517
squashing, definition of, 764
Squid proxy server
caches supported by, 473–474
configuring cache memory in, 474
configuring client browsers for, 468–469
creating ACLs (access control lists) for, 470
features of, 467–468
logs maintained by, 474
and reverse proxy cache, 474–475
security in, 470–473
using ICPs (Internet Cache Protocols) with, 473–474
and web server acceleration, 474–475
website for, 12
squid.conf file
default entries in, 471
http_access options in, 472
location and contents of, 469–470
SquirrelMail mail client, features of, 270–271

.src.rpm extension, meaning of, 220
SSH (Secure Shell), implementations of, 359. *See also* OpenSSH
ssh client, features of, 365–366
SSH clients
    scp, 366–367
    sftp and sftp-server, 367
    ssh, 365–366
ssh command, issuing, 309–310
SSH configuration files
    descriptions of, 364
    location of, 368
SSH keys, creating with ssh-keygen, 363–364
SSIs (server-side includes)
    AddHandler directive used with, 462
    AddType directive used with, 462
    features of, 462–463
    Options directive used with, 462
    XBitHack directive used with, 462
SSL (Secure Sockets Layer)
    using with Sendmal, 496
    using with web servers, 464–466
standalone server, function of, 406
standard error
    appending to files, 52
    redirecting, 44
    redirecting and piping, 51–52
standard input
    explanation of, 47
    receiving data from, 50
standard output
    explanation of, 47
    redirecting, 48–50
StarOffice software, website for, 12, 238, 240
StartServer directive, using with Apache web server, 451
startx command, issuing, 24, 156, 167
state extension, using with packets, 383
states, specifying for connection tracking, 383. *See also* runlevels
steganography, definition of, 323
sticky bit permissions, setting, 569–570
storage.fdi directory, contents of, 656–657
strings
    comparing, 77–78
    comparing in TCSH shell, 83
    quoting, 68–69
    testing for equality and inequality of, 82
    testing in TCSH shell, 82
    using with shell variables, 68–69
striping level of RAID, function of, 628
Stronghold Enterprise Server, description of, 444
su command
    description of, 525
    editing, 526
    logging into root from, 524
    using in SELinux, 329
subnetting, explanation of, 716
subshells
    generation of, 73
    referencing variables in, 74–75
substitution command, using in C shell history, 61
Subversion method, function of, 235
sudo command, description of, 525
sudo command, issuing, 525–526
Sun, website for, 289
Sun Java System, description of, 444
Sun Java website, URL for, 13
superusers, capabilities of, 523–524
swat configuration file, using with xinetd, 409
swat file, displaying in xinetd.d directory, 417
switch control structure
    effect of, 81
    using in TCSH, 84–86
switcher, using with GNOME, 20
Sybase database, features of and website for, 246–247
Sybase database software, URL for, 12
symbolic links
    for CD/DVD readers, 648
    device files as, 645–647
    function of, 130–131
SYMLINK rules, creating, 649–650
/sys file system, querying, 650–652
sysfs file system, function of, 589–590, 639–641
syslog.conf file
    effect of changes on, 537
    entries in, 539–540
    example of, 541
syslogd daemon
    actions and users in, 540–541
    facilities, priorities, and operators for, 539–540
    function of, 537
system activity information, displaying, 544
system administrator, capabilities of, 523–524
system devices, obtaining information about, 639–641

system directories
contents of, 119
in FHS (Filesystem Hierarchy Standard), 585–587
functions of, 120, 536–537
system information, displaying with ruptime command, 308
system initialization files, location of, 396
system logs, location of, 537
System menu in GNOME, capabilities of, 23
system process number, using with jobs, 53
system runlevels. *See* runlevels
system startup files, rc.sysinit and rc.local, 401
System Tap diagnostic tool, features of, 544
System V init script, tags for, 413
SysV Init, starting servers with, 403
sysv-rc-conf, managing services with, 410–412

T

Tables
Amanda commands, 696
Apache web server files and directories, 446
Apache-related websites, 445
backup resources, 694
BASH shell control structures, 79
BASH shell special features, 92
BASH shell test operators, 78
C shell history commands, 58
chage command options, 557
chkconfig options, 408
CIDR IPv4 network masks, 715
command line editing operations, 38
command line text completion commands, 40
cron commands, tools, files and directories, 530
CUPS administrative tools, 511
CUPS configuration files, 508
CUPS print clients, 509
database management systems, 246
database resources, 515
desktop editors, 249
device name prefixes, 591
device resources, 640
device symbolic links, 647
DHCP declarations, parameters, and options, 752–753
directory commands, 122
distributed file systems, 778
dump utility options, 699
editors, 249
edquota options, 572
/etc/exports options, 763
/etc/init/d service scripts, 404
fdisk commands, 608
file expansion characters, 44
file system directories, 585
file system types, 597
find command, 127
FTP clients, 291
ftp program commands, 295
FTP servers, 424
GDM configuration files and directories, 166
GFS tools, 782
GNOME configuration directories, 194
GNOME desktop menu, 176
GNOME mail clients, 269
GNOME Office, 243
GNOME resources, 170
GnuPG commands and options, 316–317
graphics tools for Linux, 258
gzip options, 139
HAL storage policies, 655
history commands and history events, 41
host.conf resolver options, 727
ICMP packets, 381
ifconfig options, 730
information resources, 29
init script functions, 413
INN configuration files, 513
IPsec resources, 350
IPsec tools, 351
IPtables commands, 377
IPtables options, 378–379
IPtables targets, 376
IPv4 local network IP addresses, 718
IPv6 format prefixes, 722
Java packages and Java web applications, 288
job management operations, 53
KDE file manager keyboard shortcuts, 211
KDE installation directories, 216
KDE websites, 198
kernel make command compiling options, 681
kernel module commands, 665
KOffice applications, 242
Linux distributions and kernel sites, 5

Linux Documentation Project (LDP), 14
Linux information and news sites, 14
Linux programming, 13
listing, displaying, and printing files, 121
LVM commands, 618
mail clients, 266
mdadm -create options, 633
mdadm modes, 630
mdadm.conf options, 631
mkfs options, 609
mount command, 602
mount options for file systems, 599
Mozilla resources, 284
MTAs (mail transfer agents), 478
multimedia and sound applications, 259
MySQL commands, 519
Nautilus File Manager menu, 181
Nautilus file pop-up menu, 183
Netfilter built-in chains, 377
Network Manager parameters, 736
network security applications, 374
network server and security software, 12
network tools, 302
newsreaders, 277
NFS mount options, 769
NSS configuration services, 728
NSS-supported files, 728
office suites, 238
OpenOffice.org applications, 239
partition and file system creation tools, 607
performance tools, 543
permissions for files and directories, 564
PGP (Pretty Good Privacy) sites, 314
PostScript, PDF, and DVI viewers, 244
print resources, 504
/proc device information files, 641
/proc subdirectories and files, 589
prompt codes, 100
query options for installed software, 225
quota options, 573
RAID levels for Linux software, 627
redirection shell operations, 48–50
remote access commands, 307
restore interactive mode shell commands, 703
restore operations and options, 702
routing table entries, 732
RPM discrepancy codes, 227
RPM options, 223–224
RPM package query options, 225
runlevels, 532
runlevels for Red Hat and Debian, 405
SELinux management tools, 333
SELinux policy configuration files, 342
SELinux resources, 328
Sendmail access actions, 496
Sendmail features, 489–490
Sendmail files and directories, 486
shell configuration files, 91
shell invocation command names, 90
shell symbols, 44
shell variables, 95
shells, 36
shutdown options, 535
software package file extensions, 220
sound devices, 662
SQL commands, 516
Squid ACL options, 471
Squid protocols, 468
SSH and Kerberos resources, 360
SSH configuration files, 364
SSH tools, 362
subnets and masks, 717
system administration tools, 525
system configuration files and directories, 538
system directories, 120, 536, 586
system environment variables used by shells, 96
system startup files and directories, 402
System V init script tags, 413
talk and messenger clients, 303
tar file archives, 135
TCP wrapper wildcards, 421
TCP/IP configuration addresses and files, 724
TCP/IP protocol development groups, 708
TCP/IP protocol suite, 709
TCSH conditional control structures, 84
TCSH loop control structures, 86
TCSH test expression operators, 83
third-party Linux software archives, repositories, and links, 11
udev rule keys, 645
udev substitution codes, 646
Usenet newsgroups, 15
user and group management tools, 557
user configuration files, 552
useradd and usermod options, 558
/usr directories in FHS, 587

Tables (*Continued*)
/var subdirectories, 588
Vi editor commands, 251–252
video and DVD projects and applications, 262
video and TV device drivers, 663
vsftpd files, 433
vsftpd.conf configuration options, 431
web file types, 283
web protocols, 282
wvdial variables, 738
X Window System application configuration options, 156
X Window System commands, 157
X Window System configuration files, 161
X Window System configuration tools, 148
XDM configuration files and directories, 164
xinetd attributes, 419–420
Xorg directories, 147
Z shell history, 64
Talk utility, features of, 305
TANGO, relationship to GNOME, 170
tape, archiving to, 138
tar archives, extracting with KDE file manager, 210
tar command
issuing, 134
using c option with, 135
using f option with, 134
using u option with, 136
using x extension with, 230
using z option with, 137, 140
.tar extension, meaning of, 220
tar file members, compressing, 140
tar utility
archiving files and devices with, 134–138
archiving to floppies with, 137
archiving to tape with, 138
compressing archives with, 137–138
creating archives with, 134–136
decompressing and extracting software with, 228–229
displaying archive contents with, 134
extracting archives with, 136
updating archives with, 136–137
.tar.bz2 extension, meaning of, 220
.tar.gz extension, meaning of, 137, 139, 220
tasks
organizing scheduled tasks, 529
scheduling with cron, 527–528
TCP wrappers, using with xinetd services, 421–422
tcpd daemon, invoking, 422
tcpdump tool, features of, 741
TCP/IP (Transmission Control Protocol/Internet Protocol), function of, 707–709
TCP/IP configuration files
/etc/hosts, 723–725
/etc/protocols, 725
/etc/resolv.conf, 725
/etc/services, 725
TCP/IP network addresses
broadcast addresses, 719
and CIDR (classless interdomain routing), 714–717
class-based IP addressing, 712–713
gateway addresses, 719
IPv4 network addresses, 712
name server addresses, 719–720
and netmasks, 713–714
TCSH shell. *See also* shells
argv array in, 72–73
case of variables in, 109
cdpath variable in, 110
command line completion in, 62
conditional control structures in, 82
configuration files for, 91
control structures in, 81–88
defining variables in, 108
environment variables in, 76
features of, 62, 89
history editing in, 62–63
history variable in, 111
if control structures in, 83
if-then structure in, 84
initialization files in, 111–114
loops in, 86–88
mail variable in, 111
prompt variables in, 109–110
relationship to C shell, 57
savhistory variable in, 111
script capabilities of, 66
switch structure in, 85–86
test expression operators in, 83
testing strings in, 82
using alias command with, 107–108
using echo command in, 108

using ignoreeof command with, 108
website, 36
.tcshrc initialization file in TCSH
description of, 111
function of, 112–113
TE (Type Enforcement), use in SELinux, 328
.te files, using in SELinux, 340–341
telinit command
changing runlevels with, 149, 533–534
description of, 525
powering down systems with, 534
Telnet sessions, prohibition of su command in, 525
Telnet utility, invoking, 306
termcap file, contents of, 661
terminal devices
initializing with tset command, 661
installing and managing, 660–661
terminal identifier (TTY), significance of, 55–56
terminal window, accessing, 27–28
test command, issuing, 77–78
TeX typesetting tool, availability of, 244
text, adding in BASH shell, 37
TGS (ticket-granting server), role in Kerberos, 369–371
TGT (ticket-granting ticket), role in Kerberos, 369–370
themes
customizing in GNOME, 24–25
managing in KDE, 25
selecting in GNOME, 172
third-party repositories, downloading software from, 11
Thompson, Ken, 7
ThreadsPerChild directive, using with Apache web server, 451
thumbnail image viewer, availability of, 257
Thunderbird mail client
enabling LDAP on, 580
features of, 268–269
ticket-granting server (TGS), role in Kerberos, 369–371
ticket-granting ticket (TGT), role in Kerberos, 369–370
tickets
destroying, 371
listing in Kerberos, 369
tilde (~), using with BASH automatic completions, 39
Timeout directive, using with Apache web server, 449
-title argument, using in X Window System, 156
TOG (The Open Group) website, accessing, 146
.torrent extension, meaning of, 220
torrents, starting, 221
Torvalds, Linus, 7
TOS targets, using, 385
Totem
features of, 263
website for, 262
traceroute network tool, 301–304
transfer protocol, role in URLs, 282
TransferLog directive, using with Apache web server, 458
Transmission Control Protocol/Internet Protocol (TCP/IP), function of, 707–709
tree structure, explanation of, 117
Tripwire, implementing intrusion detection with, 325
Trolltech website, accessing, 198
tset command, placement of, 661
TSIG key, using with DHCP dynamic DNS updates, 756
TTY (terminal identifier), significance of, 55–56
tunnel mode, using with IPsec, 356–357
tunneling, implementing in OpenSSH, 367–368
Turbo Linux, URL for, 5
Tux web server, features of, 443–444
TV devices, installing, 663
TV players, availability of, 263
tvtime, website for, 262
tvtime TV player, features of, 263
twm window manager, starting, 167–168
Type Enforcement (TE), use in SELinux, 328
-type option, using with find command, 126
TypesConfig directive, using with Apache web server, 455
.tz extension, meaning of, 220

U

Ubuntu Linux, URL for, 5
udev (user devices) tool, using, 590
udev hotplug tool
configuring, 642–643
function of, 641–642
udev rules
creating, 648–649
relationship to device names, 643–645

udevinfo command, issuing, 650–652
umask, setting permission defaults with, 570–571
UML (user-mode Linux), function of, 674
umount command, using with file systems, 603
unalias command, using with TCSH shell, 108
uncompress command, using, 140
unicast address, explanation of, 721–722
Universal Resource Locators (URLs), components of, 282
University of Washington POP and IMAP servers, website for, 499
Unix
    history of, 7
    versus Linux, 4
unrar tool, downloading, 133
unset command
    using with shell variables, 68
    using with TCSH shell, 108
unzip command, issuing, 141
update-rc.d tool, using with runlevel links, 411–412
upgrade command, using with Debian, 228
URL pathnames, directives used with, 453–454. *See also* pathnames
URLs. *See* websites
URLs (Universal Resource Locators), components of, 282
USB drives, accessing from KDE desktop, 208
UseCanonicalName directive, using with Apache web server, 460–461
Usenet articles, reading with newsreaders, 277
Usenet news
    accessing, 276
    description of, 275
    newsfeeds for, 278
    using news transport agents with, 278–279
Usenet newsgroups, availability of, 15
user chains, defining, 393
user configuration files, paths for, 552
user environments, managing, 554–557
user interface, relationship to operating system, 6
user login sessions, managing with GDM, 164–166
user logins, managing with KDM, 166–167
user passwords, controlling, 556
user private groups, creating, 560
user shell
    disabling special characters in, 109
    generation of, 73, 94
User Switcher tool, using, 20
useradd command, issuing, 558–559
user-defined chains, creating for IPtables, 380–381
user-mode Linux (UML), function of, 674
usernames
    entering, 21
    prompt code for, 100
    representing in BASH shell, 39
userod command, issuing, 559
users
    adding and removing, 557–559
    adding in SELinux, 345–346
    identifying when unmounting file systems, 603
    listing with who command, 303
    switching, 524
users logged on, determining, 553
users-admin tool, features of, 551–552
/usr directory in FHS, contents of, 587

V

/var directory in FHS, contents of, 588
variable values, using in shells and subshells, 74
variables
    assigning results of Linux commands to, 70
    case of, 109
    defining in shell scripts, 73
    defining in TCSH shell, 108
    referencing in subshells, 74–75
    representing in BASH shell, 39
/var/log directory, contents of, 537
very secure FTP server, URL for, 12
Very Secure FTP Server (vsftpd). *See* vsftpd (Very Secure FTP Server)
vgcreate command, using with LVM groups, 619
vgextend command, using with LVM groups, 620
Vi editor, using with history events, 42
video applications, and DVD players, 262–263
video devices, installing, 663
VideoLAN, website for, 262
vim command, executing, 253
Vim editor, features of, 250–254

virt-install script, using, 690–691
virt-manager in Red Hat, features of, 686–687
virtual domains, defining for Sendmail, 496
virtual hosting
    on Apache web server, 458–462
    relationship to vsftpd, 434–435
Virtual Machine Manager, features of, 686–687
VirtualDocumentRoot directive, using with Apache web server, 460
VirtualHost directive, using with ProFTPD, 440–442
virtualization methods
    availability of, 685–686
    KVM (Kernel-based Virtualization Machine), 687–688
    virt-manager in Red Hat, 686–687
    Xen Virtualization, 688–692
VirtualScriptAlias directive, using with Apache web server, 460
virtusertable.db file, generating for Sendmail, 496
visudo command, issuing, 526
vmliniz file, explanation of, 675
vmstat command, effect of, 544
VMware, website for, 239
VoIP applications, Ekiga, 304–305
VRFY option, disabling in Sendmail, 498
vsftpd (Very Secure FTP Server)
    allowing anonymous user permissions in, 432
    allowing messages in, 432
    command access in, 434
    configuring, 430
    connection time limits in, 432
    denying access in, 433
    enabling local user permissions in, 430–432
    enabling standalone and login access in, 430
    implementing virtual users in, 435
    logging in, 433
    running, 429–430
    user access in, 433
    user authentication in, 434
    user restrictions in, 434
    virtual hosting in, 434–435

W

w (write) permission, executing, 562, 566
.wav web file type, description of, 283
web browsers
    ELinks, 286
    features of, 282–283
    Firefox, 284–285
    in GNOME, 286
    K Desktop file manager window, 285–286
    Lynx, 286
    Mozilla framework, 283–284
web clients, and URL addresses, 282
web files, types of, 283
web pages, creating, 286–287
web servers
    accessing, 286
    Apache, 444–448
    Apache-SSL, 444
    lighthttpd, 444
    NCSA web server, 444
    Netscape Enterprise Server, 444
    PHP (PHP: Hypertext Preprocessor), 463
    SSIs (server-side includes), 462–463
    and SSL (Secure Sockets Layer), 464–466
    Stronghold Enterprise Server, 444
    Sun Java System, 444
    Tux, 443–444
    Zope application server, 444
Webalizer tool, generating web-log reports with, 457
web browser–based FTP, using Firefox for, 291–292
websites
    AbiSource project, 243
    Apache Jakarta Project, 445
    Apache web server, 12
    Apache-SSL, 444
    backup resources, 694
    BASH shell, 36
    Blackdown project, 287
    cedega commercial drivers, 31
    Cluster Project Page site, 780
    Courier-IMAP server, 500
    creating, 286–287
    CrossOver Office, 238
    Cyrus IMAP server, 500
    database management systems, 246
    DB2 database software, 12
    Emacs mail clients, 271
    Fluendo, 258
    GFS (Global File System), 780
    GNOME (GNU Network Object Model Environment), 170
    GNOME applications, 11
    GNOME developers website, 13

websites (*Continued*)
GNOME Keyring Manager, 317
GNOME Office, 238
GNOME Office software, 12
GNOME themes, 172
GNU archive, 11
GNU Java Compiler, 288
GNU SQL database software, 12
GnuCash finance application, 244
GnuPG (GNU Privacy Guard), 322
gPhoto project, 257
gStreamer, 259
gstreamer multimedia codecs and plugins, 11
IBM DB2 database software, 12
IMAP and POP servers, 499–500
Internet Software Consortium, 12
IP Tables firewall, 12
Jakarta Project, 288
Java applications, 288
Java repository, 11
JPackage Project, 287
KDE developers library, 13
KDE software repository, 11
KDE websites, 198
Kerberized network tools, 371
Kerberos authentication protocol, description of, 368–369
Kerberos network authentication protocol, 12
Kerberos resources, 360
kernel references, 672
KOffice software, 12
KOffice Suite for KDE, 238
Leafnode news server, 515
lighthttpd, 444
Linux compilers and tools (gcc), 13
Linux distributions and kernel, 5
Linux Foundation, 8
Linux Game Tome, 11
Linux games, 11
mail clients for GNOME, 269
Majordomo, 275
Mozilla project, 268–269
MySQL database software, 12
NCSA web server, 444
Netscape Enterprise Server, 444
network security applications, 374
Ogg Vorbis compression for music files, 261
Open Secure Shell, 12
OpenOffice, 238
OpenOffice software, 12
OpenPGP Public Keyserver project, 319
OpenSSL, 465
Oracle database software, 12
PCHDTV video card, 263
performing document searches of, 520
Perl, 13
PGP (Pretty Good Privacy), 314
POP and IMAP servers, 499–500
Postfix mail server, 12
PostgreSQL database software, 12
ProFTPD FTP server, 12
Qpopper server, 500
RPM package repository, 11
Samba SMB server, 12
Scribus desktop publishing tool, 238
SELinux resources, 328
Sendmail mail server, 12
shells, 36
SourceForge, 9, 11
Squid proxy server, 12, 467
SSH Communications Security, 359
StarOffice, 240
StarOffice software, 12
StarOffice Suite, 238
Stronghold Enterprise Server, 444
Sun, 289
Sun Java System, 444
Sun Java website, 13
Sybase database software, 12
Trolltech, 198
University of Washington POP and IMAP servers, 499
very secure FTP server, 12
VideoLAN, 262
VMware, 239
Wine, 238
Xen Virtualization Kernel, 688
XFree86 project, 145
Xorg, 146
X.org Foundation, 145
Zope application server, 444
Welsh, Matt, 13
wget tool, accessing web and FTP sites with, 293
while control structure
effect of, 81
test expressions used with, 82
using in TCSH shell, 86–87

who command, listing online users with, 302–303
whois network tool, features of, 302
window managers, using GNOME desktop with, 175–176
Windows applications, installing with Wine, 32
Windows Configuration tool, installing with, 32
Windows games, playing on Linux, 31
Windows network access, setting up, 30–31
Windows partitions, mounting, 600–601
Windows software, running on Linux, 31–32
Windows systems, browsing on GNOME, 177
Wine, website for, 238
wine command, issuing, 32
Wine Windows compatibility layer
    installing, 31
    setting up, 32
wireless networking, with Network Manager, 733–737
Wireless Tools, features of, 735
Wireshark network protocol analyzer, features of, 739–741
wireshark network tool, features of, 302
word designators, using in Z shell, 63
words
    changing parts of, 57
    erasing, 57
    referencing in events, 60
write (w) permission, executing, 562, 566
wvdial program, implementing command line PPP access with, 737–738

X

X. *See* X Window System
x (execute) permission
    explanation of, 562
    invoking, 566
    using binary masks with, 567–568
X applications, invoking, 149
X commands, examples of, 160
X configuration, obtaining description of, 158
X Display Manager (XDM)
    availability of, 160
    features of, 163–164
X displays, managing with XDM, 163–164
X interface, configuring, 146
X protocol, development of, 146
X servers
    and port security, 382
    using with X Window System, 148
X Session manager (xsm), features of, 163
X utilities, downloading, 146
X Window HOWTO documents, consulting, 149
X Window System
    commands in, 157
    configuration of, 145
    configuration tools for, 148
    controlling display of root window in, 156
    executing configuration files in, 156
    font support for, 158
    -geometry argument in, 155
    graphic configurations in, 158
    graphic programs for, 257
    setting background color in, 156
    setting fonts in, 156
    setting foreground color in, 156
    starting, 167
    -title argument in, 156
X Window System applications, starting, 155
X Window System server, installing, 145–146
X Window System sessions, starting, 166–167
X11. *See* X Window System
x264 codec, features of, 263
.Xauthority file contents of, 161
Xbase database, features of and website for, 246, 248
XBitHack directive, using with SSIs (server-side includes), 462
.Xclients file, contents of, 156
xconfig tool, using with kernels, 678–679
.Xdefaults file, accessing, 158
XDM (X Display Manager)
    availability of, 160
    features of, 163–164
Xen Virtualization, implementing, 688–692
Xerm window, starting, 167–168
XFce4 desktop, features of, 22
Xfig drawing program, features of, 257
XFree86 project website, accessing, 145
XFS font server, configuration of, 158
xine
    features of, 263
    website for, 262
xinetd (Extended Internet Services Daemon), function of, 415–416
xinetd attributes, using, 418–420
xinetd services
    configuration files for, 417
    configuring, 418

xinetd services (*Continued*)
configuring for use by chkconfig, 409–410
disabling and enabling, 418, 420–421
enabling and disabling, 409
and network security, 417
starting and stopping, 416
swat configuration file for, 409
using, 406–407
using disable attribute with, 420–421
using TCP wrappers with, 421–422
xinetd.conf file, contents of, 416
xinetd.d directory, contents of, 417
xinit command, starting X Window System with, 167
.xinitrc scripts, creating, 167
Xload command, effect of, 544
xm, managing Xen Virtual Machines with, 692
xmkmf, use of, 232
Xmodmap file, contents of, 160
Xmorph program, features of, 257
xmtr network tool, features of, 302
Xorg
directories for, 147
libraries available to, 147
location of applications and servers for, 147–148
location of configuration files for, 147
Man pages for, 147, 149
server used by, 147
updating, 147
Xorg configuration
of Device section, 154
of Files section, 151
of flags, 152
of Input Device section, 152–153
of Module section, 152
of Monitor section, 153–154
of Mouse section, 153
of multiple monitors, 155
of Screen section, 150–151
of ServerLayout section, 154–155
X.org Foundation website, accessing, 145
Xorg server, installing, 146
xorgcfg tool, using, 149
xorgconfig tool, using, 149
Xpaint program, features of, 257
xrdb command, effect of, 158
.Xresources file, entries in, 158–159
xrog configuration files, testing, 149
Xsession script
contents of, 162
environments listed in, 163
invoking, 162
xset command, effect of, 160
xsetroot command, effect of, 160
xsm (X Session manager), features of, 163
Xterm file, contents of, 158
XviD
features of, 264
website for, 262

Z

.Z extension, meaning of, 220
Z shell. *See also* shells
configuration files for, 91
features of, 63, 89
history commands in, 64
initialization and configuration files for, 90
website, 36
word designators in, 64
Zero Configuration Networking (zeroconf), capabilities of, 710–711
zip command, using -r option with, 140–141
Zip utility, compressing files with, 140–141
Zope application server, description of, 444