

ROSS BRUNSON
SEAN WALBERG



Cert Guide

Learn, prepare, and practice for exam success



CompTIA[®] Linux+/ LPIC-1

Save 10%
on Exam
Voucher

See Inside

Exams LX0-103 & LX0-104/
101-400 & 102-400



DVD INCLUDED

PEARSON IT
CERTIFICATION

FREE SAMPLE CHAPTER



SHARE WITH OTHERS

CompTIA® Linux+ / LPIC-1 Cert Guide

Ross Brunson
Sean Walberg

PEARSON

800 East 96th Street
Indianapolis, Indiana 46240 USA

CompTIA Linux+ / LPIC-1 Cert Guide (Exams LX0-103 & LX0-104/101-400 & 102-400)

Ross Brunson

Sean Walberg

Copyright © 2016 Pearson Certification

All rights reserved. No part of this book may be reproduced or transmitted in any form or by any means, electronic or mechanical, including photocopying, recording, or by any information storage and retrieval system, without written permission from the publisher, except for the inclusion of brief quotations in a review.

ISBN-13: 978-0-7897-5455-4

ISBN-10: 0-7897-5455-X

Library of Congress Control Number: 2015945796

Printed in the United States of America

First Printing December 2015

Trademarks

All terms mentioned in this book that are known to be trademarks or service marks have been appropriately capitalized. Pearson cannot attest to the accuracy of this information. Use of a term in this book should not be regarded as affecting the validity of any trademark or service mark.

Warning and Disclaimer

This book is designed to provide information about Linux. Every effort has been made to make this book as complete and as accurate as possible, but no warranty or fitness is implied. The information is provided on an “as is” basis. The authors and publisher shall have neither liability nor responsibility to any person or entity with respect to any loss or damages arising from the information contained in this book or from the use of the DVD or programs that may accompany it. The opinions expressed in this book belong to the authors and are not necessarily those of Pearson IT Certification.

Special Sales

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

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

For questions about sales outside the U.S., please contact international@pearsoned.com.

Publisher

Paul Boger

Associate Publisher

Dave Dusthimer

Executive Editor

Mary Beth Ray

Development Editor

Eleanor C. Bru

Managing Editor

Sandra Schroeder

Project Editor

Seth Kerney

Copy Editor

Geneil Breeze

Indexer

Tim Wright

Proofreader

Brad Herriman

Technical Editors

Ted Jordan

William “Bo” Rothwell

Publishing Coordinator

Vanessa Evans

Multimedia Developer

Lisa Matthews

Interior Designer

Mark Shirar

Cover Designer

Alan Clements

Composition

Trina Wurst

Contents at a Glance

Introduction

CHAPTER 1	Installing Linux	3
CHAPTER 2	Boot Process and Runlevels	27
CHAPTER 3	Package Install and Management	51
CHAPTER 4	Basic Command Line Usage	91
CHAPTER 5	File Management	109
CHAPTER 6	Text Processing/Advanced Command Line	151
CHAPTER 7	Process Management	193
CHAPTER 8	Editing Text	219
CHAPTER 9	Partitions and Filesystems	241
CHAPTER 10	Permissions and Ownership	289
CHAPTER 11	Customizing Shell Environments	311
CHAPTER 12	Shell Scripting	339
CHAPTER 13	Basic SQL Management	365
CHAPTER 14	Configuring User Interfaces and Desktops	397
CHAPTER 15	Managing Users and Groups	419
CHAPTER 16	Schedule and Automate Tasks	445
CHAPTER 17	Configuring Print and Email Services	465
CHAPTER 18	Logging and Time Services	497
CHAPTER 19	Networking Fundamentals	529
CHAPTER 20	System Security	571
CHAPTER 21	Final Preparation	603
APPENDIX A	Answers to the “Do I Know This Already?” Quizzes and Review Questions	619
GLOSSARY		659
INDEX		693
ON THE DVD		
APPENDIX B	Study Planner	

Contents

Chapter 1	Installing Linux	3
	“Do I Know This Already?” Quiz	3
	Understanding Your Hardware	6
	Peripheral Compatibility	7
	Enumerating Your Peripherals	7
	The Proc Filesystem	8
	<i>Friends of procs</i>	10
	<i>Dealing with Integrated Peripherals</i>	10
	Laying Out the Hard Drive	11
	Partitions and Devices	11
	The Root Filesystem	12
	Logical Volume Manager (LVM)	14
	Commonly Used Mounts	16
	Swap Files	16
	Working with Boot Managers	17
	GRUB Legacy	17
	GRUB2	18
	<i>Installing GRUB2</i>	18
	<i>Using the GRUB2 Command Line</i>	19
	<i>Configuring GRUB2</i>	20
	Summary	21
	Exam Preparation Tasks	22
	Review All Key Topics	22
	Define Key Terms	22
	Review Questions	23
Chapter 2	Boot Process and Runlevels	27
	“Do I Know This Already?” Quiz	27
	The Linux Boot Process	30
	What Is the Boot Process?	30
	Boot Loaders	30
	Common Commands at Boot Time	32
	Boot Sequence from BIOS to Fully Running System	33
	SysVinit	33
	systemd	34
	Units in systemd	36
	systemd Targets and Runlevels	37
	Wants and Requires	38
	Booting with systemd	38
	Upstart	39
	<i>Managing System Runlevels</i>	40
	<i>Determining the Default Runlevel</i>	40

	Setting the Default Runlevels	41
	Changing Runlevels	41
	Shut Down and Reboot from the Command Line	42
	Alerting Users	43
	Properly Terminating Processes	44
	kill and killall	44
	Reloading or “Hanging Up” Processes	45
	Logging Boot Events	45
	Exam Preparation Tasks	46
	Review All Key Topics	46
	Define Key Terms	47
	Review Questions	47
Chapter 3	Package Install and Management	51
	“Do I Know This Already?” Quiz	51
	Software in a Linux System	54
	Shared Libraries	54
	Working with Shared Libraries	56
	Determining Required Libraries	56
	<i>Looking for Libraries in Other Places</i>	57
	Debian Package Management	58
	Managing Local Debian Packages	58
	<i>Installing Packages with dpkg</i>	59
	<i>Removing Packages</i>	60
	<i>Dependency Issues</i>	60
	<i>Querying Packages</i>	61
	<i>Reconfiguring Packages</i>	63
	Using Remote Repositories	64
	<i>Installing Remote Packages</i>	64
	<i>Working with the Cache</i>	65
	<i>Upgrading the System</i>	66
	<i>Removing Packages</i>	67
	<i>Graphical Managers</i>	67
	RPM and YUM Package Management	68
	The RPM Database	68
	<i>RPM Package Files</i>	69
	<i>Package Name Conventions</i>	69
	The rpm Command	70
	<i>Validation of Packages</i>	71
	<i>Installation of Packages</i>	72
	<i>Additional Installation Options</i>	72
	<i>Verifying a Package’s Integrity</i>	73
	<i>Freshening Versus Upgrading</i>	74
	<i>Removing Packages</i>	75
	<i>Other Removal Options</i>	76

	Querying Packages	77
	Package Management with YUM	80
	<i>Installing Packages</i>	80
	<i>Fetching Updates</i>	83
	<i>Finding Packages to Install</i>	83
	Configuring Yum	84
	Summary	86
	Exam Preparation Tasks	86
	Review All Key Topics	86
	Define Key Terms	87
	Review Questions	88
Chapter 4	Basic Command Line Usage	91
	“Do I Know This Already?” Quiz	91
	What Is a Shell?	93
	Global and User Settings	94
	Sourcing Versus Executing	94
	<i>Executing a Script</i>	94
	Sourcing a Script	94
	A Login Shell Session	95
	A Non-Login Shell Session	96
	Using the Command Line	97
	There Are Commands and, Well, Commands	97
	Structuring Commands	98
	Breaking Long Command Lines	98
	Command Completion	99
	Special Characters in the Shell	99
	Controlling Command Execution	100
	Possible Exit Statuses	100
	Environment Variables and Settings	101
	The Path	102
	Getting \$HOME	102
	bash’s History Feature	103
	Important History Variables	104
	Setting Options in bash	104
	Important bash Options	104
	Exam Preparation Tasks	105
	Review All Key Topics	105
	Define Key Terms	106
	Review Questions	106
Chapter 5	File Management	109
	“Do I Know This Already?” Quiz	109
	Filesystem Overview	112

What Belongs Where	112	
<i>The Root of the System</i>	112	
<i>Classifying Data</i>	113	
<i>Where Programs Live</i>	114	
File Management Commands	114	
Tips for Working with Linux Files	115	
Basic Navigation	115	
Advanced Navigation	116	
Listing Files and Directories	116	
Determining File Types	118	
Touching Files	120	
Copying Files and Directories	121	
Moving Objects	123	
Transforming Data Formats	126	
Creating and Removing Directories	127	
Removing Objects	128	
Where Are Those Files?	128	
Locating Files with Locate	128	
Finding Files	130	
Which Command Will Run?	132	
Researching a Command	132	
Linking Files	133	
<i>Symbolic Links</i>	134	
<i>Hard Links</i>	135	
Backup Commands	136	
Using tar	137	
Taking Pity on the Unarchiver	139	
<i>Useful Creation Options</i>	140	
<i>Listing Archive Files</i>	140	
<i>Using cpio</i>	141	
Compression Utilities	143	
Summary	144	
Exam Preparation Tasks	144	
Review All Key Topics	144	
Define Key Terms	145	
Review Questions	146	
Chapter 6	Text Processing/Advanced Command Line	151
“Do I Know This Already?” Quiz	151	
Working with Input/Output Streams	154	
Standard In	154	
Standard Out	154	
Standard Error	155	
Redirection of Streams	156	

	Redirecting Standard Input	157
	Redirecting Standard Output	157
	Redirecting Standard Error	157
	Redirection Redux	158
	Pipes	159
	Executing Multiple Commands	161
	Multiple Command Operators	161
	Command Substitution	162
	Splitting and Processing Streams	163
	Splitting Streams with the tee Command	163
	Processing Output with the xargs Command	163
	Filters	165
	Sorting	165
	Numbering Lines	166
	Tabs	167
	Cutting Columns	168
	Pasting and Joining	168
	Unique Data	169
	Heads or Tails?	170
	Splitting Files	172
	When cat Goes Backward	173
	Viewing Binary Files Safely	173
	Formatting Commands	174
	Translating Files	175
	He sed , She sed	176
	Getting a grep	178
	<i>Examples of Using grep</i>	179
	Expanding grep with egrep and fgrep	184
	Using Regular Expressions and grep	185
	Summary	188
	Exam Preparation Tasks	188
	Review All Key Topics	188
	Define Key Terms	189
	Review Questions	189
Chapter 7	Process Management	193
	“Do I Know This Already?” Quiz	193
	Managing Processes	196
	Viewing Processes	196
	What’s the Diff?	197
	The free Command	198
	Blocks and Buffers	199
	Pages, Slabs, and Caches	199

	Interpreting Displayed Information from free	200
	System Uptime	201
	Sending Signals to Processes	202
	Killing Processes by PID	203
	Killing Processes by Other Criteria	204
	Job Control	205
	Managing Process Priorities	207
	Leaving Programs Running after Logout	209
	Using screen for Multiple Console Sessions	210
	<i>Taking Control-a of screen Windows</i>	211
	<i>Creating Windows in screen</i>	211
	<i>Detaching and Reattaching from screen</i>	212
	<i>Locking Your Console via screen</i>	213
	Summary	213
	Exam Preparation Tasks	214
	Review All Key Topics	214
	Define Key Terms	215
	Review Questions	215
Chapter 8	Editing Text	219
	“Do I Know This Already?” Quiz	219
	A Tour of the vim Editor	222
	The Message Line	222
	Editing in vi	222
	<i>Opening a File for Editing</i>	223
	<i>Navigating Within a File</i>	224
	<i>Force Multipliers</i>	225
	<i>Undo Operations</i>	225
	<i>Saving Files</i>	226
	<i>Quitting vi</i>	226
	<i>Changing or Replacing Text</i>	227
	<i>Deleting Text and Lines</i>	227
	<i>The Cut, Copy, and Paste Commands</i>	228
	<i>Named and Unnamed Buffers</i>	229
	Searching in vi	230
	<i>Searching and Replacing</i>	231
	<i>Regular Expression Searches</i>	231
	Options in vi	232
	Advanced vi	234
	<i>Running External Commands in vi</i>	234
	<i>Joining Lines</i>	234
	<i>Split Windows</i>	234
	Exam Preparation Tasks	236
	Review All Key Topics	236
	Review Questions	236

Chapter 9 Partitions and Filesystems 241

“Do I Know This Already?” Quiz	241
Creating Partitions	244
Partitions	244
Swap	245
Disk Partitioning Tools	245
<i>fdisk</i>	246
<i>gdisk</i>	250
<i>Parted</i>	254
Filesystems	255
Filesystem Types	255
Superblocks	256
Inodes and Files	256
Inodes and Directories	258
Inodes and Disk Space	259
Creating Filesystems	260
<i>The mkfs Command</i>	260
<i>Filesystem Creation Options</i>	261
Advanced Filesystem Commands	263
<i>Filesystem Checker</i>	263
<i>Tuning Filesystems</i>	266
<i>XFS Commands</i>	267
<i>Debugging Filesystems</i>	268
<i>Mounting and Unmounting</i>	269
The Filesystem Table	270
<i>Manually Mounting Filesystems</i>	272
<i>Automatically Mounting Filesystems</i>	273
<i>Unmounting Filesystems</i>	273
Space Utilization	274
Using <i>du</i>	274
Using <i>df</i>	275
Using Disk Quotas	277
<i>Quota Commands and Files</i>	277
<i>Quota Concepts</i>	277
<i>Configuring Quotas</i>	278
<i>Hard and Soft Limits</i>	280
<i>Setting the Grace Period</i>	281
<i>Getting Quota Information</i>	281
Summary	282
Exam Preparation Tasks	282
Review All Key Topics	282
Define Key Terms	284
Review Questions	284

Chapter 10	Permissions and Ownership	289
	“Do I Know This Already?” Quiz	289
	Working with Permissions	292
	Permission Trio Bits	292
	Manipulating Permissions	294
	Numeric Mode	294
	<i>Symbolic Mode</i>	296
	Special File Permissions	297
	Special Bit Permissions	298
	Setting the SUID Bit on Files	299
	Setting the SGID Bit on Files	300
	Setting the SGID Bit on Directories	301
	Setting the Sticky Bit	302
	Finding Files by Permission	302
	Default Permissions	303
	Changing User Ownership	305
	Changing Group Ownership	306
	Summary	307
	Exam Preparation Tasks	307
	Review All Key Topics	307
	Define Key Terms	308
	Review Questions	308
Chapter 11	Customizing Shell Environments	311
	“Do I Know This Already?” Quiz	311
	Working Within the Shell	314
	Environment Variables	315
	Variable Scope	316
	<i>Setting Variables from a Child</i>	316
	Setting and Unsetting Variables	317
	Subshells	318
	<i>The env Wrapper</i>	319
	Extending the Shell	320
	Global and User Settings	320
	<i>A Login Shell Session</i>	321
	<i>A Non-Login Shell Session</i>	321
	The PATH	322
	<i>Aliases and Functions</i>	323
	<i>Functions</i>	323
	PS1	324
	<i>Adding More Dynamic Content</i>	325
	PS2	326
	Creating New Users (skeleton)	326

Localization and Internationalization	327
Time Zones	327
<i>Displaying Time</i>	328
<i>Setting Time Zones</i>	328
Character Encoding	329
Representing Locales	330
<i>Fallback Locales</i>	331
Contents of a Locale	331
How Linux Uses the Locale	332
<i>Converting Files Between Encodings</i>	334
Exam Preparation Tasks	334
Review All Key Topics	334
Define Key Terms	335
Review Questions	335
Chapter 12 Shell Scripting	339
“Do I Know This Already?” Quiz	339
Basics of Scripting	342
Running a Script	343
Good Design	343
Managing Your Scripts	344
Shell Script Commands	344
Use the Output of Another Command	344
Do Math	345
Conditions	346
Testing Files	348
An Easier Test Syntax	348
Testing Strings	349
Testing Integers	350
Combining Multiple Tests	351
Case Statements	351
Loops	353
<i>For Loops</i>	353
<i>Sequences</i>	354
<i>While Loops</i>	355
<i>Reading from stdin in a Loop</i>	356
Interacting with Other Programs	356
Returning an Error Code	357
Accepting Arguments	357
Transferring Control to Another Program	358
Exam Preparation Tasks	359
Review All Key Topics	359
Define Key Terms	360
Review Questions	360

Chapter 13	Basic SQL Management	365
	“Do I Know This Already?” Quiz	365
	Database Basics	368
	Types of Databases	368
	<i>Key-Value Databases</i>	368
	<i>Relational Databases</i>	369
	<i>Schemaless Databases</i>	370
	Learning SQL	371
	Using SQLite	371
	SQL Basics	372
	<i>Keywords Versus Data</i>	372
	Selecting Data	373
	<i>Being Choosy</i>	374
	<i>Multiple Conditions</i>	376
	<i>Sorting</i>	377
	Limiting Results	378
	Working with Multiple Tables	378
	<i>Writing Queries with Joins</i>	379
	<i>Cleaning Up the Query</i>	381
	Advanced Joins	381
	<i>Left Versus Right Joins</i>	384
	<i>Null</i>	384
	Subselects	385
	Grouping Data	386
	Inserting Data	387
	Updating Data	388
	Deleting Data	388
	Creating Tables	388
	Summary	390
	Exam Preparation Tasks	390
	Review All Key Topics	390
	Define Key Terms	391
	Review Questions	392
Chapter 14	Configuring User Interfaces and Desktops	397
	“Do I Know This Already?” Quiz	397
	Quick Overview of X	400
	How X Works	400
	<i>Window Managers</i>	401
	<i>Linux Desktops</i>	402
	The Xorg System	402
	The Xorg Configuration File	402
	Fonts in X	405
	Tuning X	406

	X Display Managers	408
	Into and Out of X	409
	Accessibility Options	410
	Sticky/Repeat Keys	410
	Slow/Bounce/Toggle Keys	411
	Mouse Keys	411
	High Contrast/Large Print Desktop Themes	412
	Screen Reader	412
	Braille Display	413
	Screen Magnifier	413
	Onscreen Keyboard	413
	Remote Clients	413
	Summary	415
	Exam Preparation Tasks	415
	Review All Key Topics	415
	Define Key Terms	415
	Review Questions	416
Chapter 15	Managing Users and Groups	419
	“Do I Know This Already?” Quiz	419
	User Account Fundamentals	422
	What Accounts Are What?	422
	<i>Normal User Accounts</i>	423
	<i>User Entries in /etc/passwd</i>	423
	<i>Special Login Files</i>	424
	Group Accounts	425
	Group Entries in /etc/group	427
	Group Passwords	427
	Adding Users and Groups	428
	Adding Users with useradd	428
	The useradd Defaults	429
	Adding Groups with groupadd	431
	Modifying Users and Groups	431
	Modifying User Accounts with usermod	431
	Modifying Groups with groupmod	432
	Removing Users and Groups	433
	Removing Users	433
	Removing Groups	434
	The Shadow Suite	435
	Encrypted Passwords and Shadow Fields	435
	shadow File Permissions	437
	Changing Accounts	437
	Aging Passwords	438
	User Variables	440

	Summary	440
	Exam Preparation Tasks	441
	Review All Key Topics	441
	Define Key Terms	442
	Review Questions	442
Chapter 16	Schedule and Automate Tasks	445
	“Do I Know This Already?” Quiz	445
	The Cron System	447
	Configuring crontabs	447
	<i>Using the crontab Command</i>	447
	<i>Matching Times</i>	448
	<i>Spelling Out Month and Day Names</i>	449
	<i>Making Multiple Matches</i>	449
	<i>Step Values</i>	450
	<i>Putting the crontab Together</i>	450
	<i>Issues About Path</i>	450
	<i>Dealing with Output</i>	451
	<i>Nicknames</i>	452
	Other Files	452
	System crontabs	453
	Convenience crontabs	454
	Restricting Access	454
	Anacron	455
	Running Ad-hoc Jobs	456
	The at Command	456
	The batch Command	458
	Summary	459
	Exam Preparation Tasks	460
	Review All Key Topics	460
	Define Key Terms	460
	Review Questions	461
Chapter 17	Configuring Print and Email Services	465
	“Do I Know This Already?” Quiz	465
	Managing Printers and Printing	467
	The Print Spooler	467
	<i>Network Printing Protocols</i>	467
	The CUPS Daemon	468
	<i>The CUPS Pipeline</i>	468
	Configuring CUPS	470
	CUPS Maintenance	474
	<i>Printer State</i>	475
	<i>Maintenance and Administration Pull-downs</i>	476

<i>Administration Menu</i>	476
<i>Jobs List</i>	477
Command Line Tools	477
<i>Legacy Tools</i>	477
<i>lp/lpr</i>	478
<i>lpstat</i>	478
<i>lpq</i>	479
<i>lprm</i>	479
CUPS Tools	480
<i>cupsaccept/cupsreject</i>	480
<i>cupsenable/cupsdisable</i>	481
<i>cupsctl</i>	481
Configuration Files	482
Troubleshooting Printing	482
<i>Try Printing from the Command Line</i>	482
<i>Was the Job Queued?</i>	483
<i>Can CUPS Send the Job to the Printer?</i>	483
<i>Turn On Debugging</i>	483
Mail Transfer Agent Basics	484
How Email Flows	484
<i>Mail User Agent</i>	484
<i>Mail Transfer Agent</i>	485
<i>The Language of Email</i>	485
Linux MTAs	486
Domain Name System	487
<i>Mail Delivery Agent</i>	488
<i>Mail Server</i>	488
Creating Aliases and Forwarding Email	489
<i>Committing Changes</i>	489
<i>Other Types of Aliases</i>	490
<i>User-Defined Forwarding</i>	491
Managing Queues	491
Summary	492
Exam Preparation Tasks	492
Review All Key Topics	492
Define Key Terms	493
Review Questions	493
Chapter 18 Logging and Time Services	497
“Do I Know This Already?” Quiz	497
Maintain System Time	500
Not One, But Two Clocks	500
Working with the System Clock	500
Working with the Hardware Clock	503

	<i>The hwclock Command</i>	503
	<i>Synchronizing Time Between Clocks</i>	504
	Network Time Protocol	504
	<i>Setting the Time from the Command Line</i>	505
	<i>The pool.ntp.org Servers</i>	505
	<i>Configuring ntpd</i>	506
	<i>Monitoring ntpd</i>	507
	System Logging	508
	systemd and syslog	508
	syslog	509
	<i>The logger Command</i>	512
	<i>Configuring syslogd</i>	512
	<i>Other syslog Implementations</i>	514
	systemd Logging	514
	<i>Querying the Log</i>	515
	<i>Configuring journald</i>	519
	Rotating Logs	520
	Configuring Log Rotation	520
	Dealing with Open Files	522
	Summary	522
	Exam Preparation Tasks	523
	Review All Key Topics	523
	Define Key Terms	524
	Review Questions	524
Chapter 19	Networking Fundamentals	529
	“Do I Know This Already?” Quiz	529
	Conceptual Overview of Networking	532
	Necessary Configuration Information	533
	IP Addresses	533
	Networks and Hosts	534
	Address Class Ranges	534
	<i>Using the Bits to Determine Class</i>	535
	Network Masks	536
	Using Default Network Masks	536
	Gateway Addresses, or “Do I Dial with the Area Code?”	537
	Broadcast Addresses	538
	Custom Network Masks	538
	Determining a Custom Network Mask	538
	Additional Protocols	541
	<i>Common Ports</i>	542
	IPv6	544

	Managing Interfaces	545
	Viewing IP Information	545
	Red Hat Interface Configuration	547
	Debian Interface Configuration	548
	Viewing and Configuring Gateway Addresses	550
	<i>Viewing the Default Gateway</i>	550
	<i>Configuring a Default Gateway</i>	550
	<i>Local Name Configuration</i>	551
	Network Configuration Utilities	553
	Network Utility Examples	554
	<i>The ifconfig Command</i>	555
	<i>The route Command</i>	555
	<i>DHCP Client Tools</i>	556
	<i>The host, getent, and dig Commands</i>	557
	<i>Hostname Utilities</i>	559
	<i>Using netstat</i>	559
	<i>The ping Command</i>	562
	<i>Using traceroute</i>	563
	<i>Using tcpdump</i>	565
	Summary	566
	Exam Preparation Tasks	566
	Review All Key Topics	566
	Define Key Terms	567
	Review Questions	567
Chapter 20	System Security	571
	“Do I Know This Already?” Quiz	571
	Gaining Access to the root Account	574
	The su Command	574
	The sudo Command	575
	Providing Services on Demand	576
	Using inetd and xinetd	576
	inetd Configuration Files	577
	xinetd Configuration Files	578
	Using TCP Wrappers for Securing Services	581
	inetd and TCP Wrappers	581
	xinetd and TCP Wrappers	582
	The hosts.allow and hosts.deny Files	582
	Wrapper Read Order	583
	Format of hosts.allow and hosts.deny	583
	Sample Configurations	583
	Using Rule Options	585

	Understanding Permission Problems	586
	Finding Files by Permissions	587
	GnuPG Keys	587
	Secure Shell	590
	SSH Components	591
	<i>Using SSH Client Utilities</i>	592
	Additional Security Features	596
	Summary	597
	Exam Preparation Tasks	598
	Review All Key Topics	598
	Define Key Terms	599
	Review Questions	599
Chapter 21	Final Preparation	603
	How to Prepare for the LPI Exams	604
	Caveat and Warning	604
	Exam Objectives	604
	Important Exam Facts	605
	Right Before Your Exam Starts	605
	How to Look at the Objectives	606
	Studying for the Exams—What to Do	608
	Machines or Virtual Machines?	609
	Studying for the Exams—What Not to Do	609
	Don't Believe Everything	610
	Don't Worry, Be Happy	610
	LPI Certifications and Distributions	610
	You Have to Install Something	611
	LPI Exam Question Types	611
	Single Answer Multiple Choice	612
	Choose Two/Choose Three	613
	Choose All That Apply	613
	Fill in the Blank	615
	Final Recommendations	616
	Summary	617
Appendix A	Answers to the “Do I Know This Already?” Quizzes and Review Questions	619
	Glossary	659
	Index	693

About the Authors

Ross Brunson has more than 20 years of experience as a Linux and Open Source trainer, training manager, and technologist and is author of the popular LPIC-1 Exam Cram (QUE Publishing).

Ross is currently senior training/certification engineer at SUSE and recently spent almost five years as the director of member services for the Linux Professional Institute, where he contributed to placing several LPI courses into the Cisco Networking Academy, conducted dozens of Train-the-Trainer sessions, and provided sales enablement support for the worldwide Master Affiliate network spanning more than 100 countries.

Ross holds a number of key IT certifications and is also author of several successful technical books and dozens of technical courses for major organizations (including the first LPI Certification Bootcamps). He is skilled at both contributing to and building community around IT products.

He lives in Paradise Valley, Montana, with his family and enjoys traveling far and wide, winter sports, and photography.

Sean Walberg has more than 20 years of experience as a Linux administrator, network engineer, and software developer. He has written extensively on Linux certification for IBM and NetDevGroup, and has contributed to other books both as an author and technical reviewer.

Sean currently works at Northfield IT and is responsible for infrastructure automation for a large professional sports league. Using tools like Ruby, shell scripts, and Chef, he automates the creation and maintenance of more than a thousand servers and the associated network infrastructure. Sean works closely with developers to scale applications to the demands of an internationally recognized series of web properties.

He lives in Northern Virginia with his wife and three sons.

About the Contributing Author

At the impressionable age of 14, **William “Bo” Rothwell** crossed paths with a TRS-80 Micro Computer System (affectionately known as a “Trash 80”). Soon after the adults responsible for Bo made the mistake of leaving him alone with the TRS-80. He immediately dismantled it and held his first computer class, showing his friends what made this “computer thing” work.

Since this experience, Bo’s passion for understanding how computers work and sharing this knowledge with others has resulted in a rewarding career in IT training. His experience includes Linux, Unix, and programming languages such as Perl, Python, Tcl, and BASH. He is the founder and president of One Course Source, an IT training organization.

About the Technical Reviewer

Ted Jordan has more than 25 years of programming, administration, and training experience in UNIX, IRIX, Solaris, and Linux. His career spans from General Motors, Silicon Graphics, to SUN. He holds the LPIC, Linux+, and SUSE Linux certifications. He is the founder and president of two successful startups, the latest being Funutation Tech Camps where he teaches kids to code computer games.

Ted lives with his family near Worcester, Massachusetts, and enjoys tennis, golf, and karaoke.

Dedications

Ross Brunson: *To my good friends, Andres and Ken, we few, we happy few. With love and respect to my wife and daughter; for putting up with my being locked in my office writing and editing while the sun shone and breezes blew. To every student/attendee/customer I’ve ever taught a Linux topic to, it’s really all for you.*

Sean Walberg: *To my amazingly beautiful and intelligent wife, Rebecca. The completion of this book happens to coincide with the start of our new adventure together, and I can think of no one else I’d like to share it with.*

Acknowledgments

Ross Brunson: To the ultimate nerd-herders, Ellie Bru and Mary Beth Ray; Ellie for her ability to take the peeping and muttering of technical geeks and make it something useful, workable, and often profound, and Mary Beth for believing in authors and technologists, even when we break her heart by blowing out deadlines and not doing what we say we will on time.

To my little brother, Leighton, who will make the most awesome history professor one day.

To Sean Walberg, who I have known and respected over several book projects, years of interactions in the LPI community, and all the way back to the Cramsession days with Qcumber and the gang.

To Bo Rothwell and Ted Jordan, awesome technical editors and great guys, as well as two of the best technical trainers it is my pleasure to know.

Sean Walberg: To the crew at Pearson, most notably Geneil, Ellie, and Mary Beth: This project is better because of your patience and input.

The technical editors, Ted and Bo, also deserve special mention. Not only did you fix my technical missteps, but your years of experience as trainers pointed out where I was using some words that were going to confuse new Linux users.

Finally, my knowledge of Linux wasn't earned alone. It came through long nights, hard work, and lots of arguing with people like Marc Caron, Hany Fahim, Patrick leMaistre, Daniel Little, Dave Rose, and of course, my co-author Ross Brunson who I'm happy to have known for more than 15 years.

We Want to Hear from You!

As the reader of this book, you are our most important critic and commentator. We value your opinion and want to know what we're doing right, what we could do better, what areas you'd like to see us publish in, and any other words of wisdom you're willing to pass our way.

We welcome your comments. You can email or write to let us know what you did or didn't like about this book—as well as what we can do to make our books better.

Please note that we cannot help you with technical problems related to the topic of this book.

When you write, please be sure to include this book's title and author as well as your name and email address. We will carefully review your comments and share them with the author and editors who worked on the book.

Email: feedback@pearsonitcertification.com

Mail: Pearson IT Certification

ATTN: Reader Feedback

800 East 96th Street

Indianapolis, IN 46240 USA

Reader Services

Register your copy of CompTIA Linux+ / LPIC-1 Cert Guide at www.pearsonitcertification.com for convenient access to downloads, updates, and corrections as they become available. To start the registration process, go to informit.com/register and log in or create an account. Enter the product ISBN (9780789754554) and click Submit. Once the process is complete, you will find any available bonus content under "Registered Products." Be sure to check the box that you would like to hear from us in order to receive exclusive discounts on future editions of this product.

Introduction

This book was written to help people learn to use Linux. Not just learning Linux by memorizing commands, but learning Linux by understanding how the parts are put together. Approaching Linux from this perspective means that you'll know where to look when you run up against something new and are better suited to handle problems as they come up. The authors of this book are experienced writers, but more importantly, are in the trenches every day.

The CompTIA Linux+ exams LX0-103 and LX0-104 and Linux Professional Institute LPIC Level 1 exams 101-400 and 102-400 (which are identical) encompass the knowledge necessary to become an entry level Linux administrator. There are certainly other books that cover this material, but this is the one that looks beyond the exam to preparing people for the Linux workforce.

You don't need to be taking either the Linux+ or LPIC exams to get use out of this book. Concepts such as filesystems, hardware, shell usage, and managing email systems are needed in the workforce, and we, as authors, have endeavored to produce a book that is just as helpful to all new Linux users.

Goals and Methods

The goal of this book is to provide a guided tour of the Linux operating system with an eye to achieving an entry-level certification at the completion of the book. Readers with no intention of writing an exam will still find this book helpful as the certification content, by design, closely maps to the skills required by a Linux administrator. The authors also hope that the examples and practical advice in this text prove valuable well after the reader is done with the book.

The Linux+ and LPIC Level 1 certification exams are broken into specific topics that build upon each other, and the book does its best to mirror those. Not only does this provide a natural progression to learning Linux, but for those who are taking the exam, allows them to focus on troublesome areas.

Linux commands and their output are interspersed with the text to provide concrete examples right next to the description. Examples, for the most part, are adaptations of real world usage rather than being contrived. And since no good Linux graybeard should take himself too seriously, the authors have done their best to inject some levity into the discussion.

Who Should Read This Book?

This book was written for people who want to learn Linux—people just getting into the information technical field, Windows administrators who want to branch out to Linux, or students looking to understand Linux. Even if you’re not taking the Linux+ or LPIC Level 1 exams you’ll find this book helpful.

The first half of the book focuses on concepts and basic command usage, while the second half turns the attention to applications found in a typical Linux environment. People looking to be more competent Linux users, as opposed to administrators, will find immense benefit in the first half of the book, but will still appreciate the view of what else can be done on Linux provided by the second half.

Managers looking for some Linux familiarity will also find this book helpful because of the abundant examples and real world applications that will help them to speak in the same language as their more technical reports.

This book was not meant just to be read and cast aside. Instead, it can be a reference for common command usage and some basic application administration.

How To Use This Book

The best way to learn Linux is to use Linux. There are many examples within the text, from simple one-line commands to reusable scripts. Find yourself a Linux distribution such as Fedora, Ubuntu, Debian, or openSUSE. They’re free and run on most hardware.

If you don’t have a spare computer on which to install Linux you can try a LiveCD, which is a bootable image that runs entirely in memory. Most distributions offer a LiveCD download. Alternatively, you can run Linux in a virtual machine with software like VirtualBox (<http://www.virtualbox.org>).

All the software shown in this book is available on the most basic of Linux distributions and does not need an extra download. However Chapter 13, “Basic SQL Management,” offers a sample database that you can use to follow the examples. To install this, download the compressed attachment from <http://www.pearsonitcertification.com/title/9780789754554>. Inside the compressed file are two database files. The first, called **lpic_basic.sqlite3**, contains the data for the first part of the chapter. The second includes the additional data for the later examples. Instructions for using the databases are found in Chapter 13.

Above all, experiment with your Linux system. Try a couple of different distributions. Run the commands in this book and see whether you can come up with your own examples. Poke around in the configuration files and explore alternative uses for the commands in this book.

How This Book Is Organized

Although you could read this book cover-to-cover, it is designed to be flexible and allow you to easily move between chapters and sections of chapters to cover only the material you need. If you do intend to read them all, the order in which they are presented is an excellent sequence.

Chapters 1 through 12 cover the following topics:

- **Chapter 1, “Installing Linux”:** This chapter teaches you the basics of how a Linux system is installed. Core topics like hard disk partitioning and dealing with hardware are the focus of this chapter.
- **Chapter 2, “Boot Process and Runlevels”:** The Linux system has a specific order in which things happen both for starting up and shutting down. This chapter discusses the way these processes work and how to make changes so that you get the services that you need on your system.
- **Chapter 3, “Package Install and Management”:** Finding, installing, and configuring software is a big part of the system administrator’s job description. This chapter walks you through the usage of both the Debian and RedHat package systems.
- **Chapter 4, “Basic Command Line Usage”:** This chapter takes you through the basics of working on the Linux command line, including running applications and some commands to orient yourself on a new system. The work here forms the basis of the next three chapters.
- **Chapter 5, “File Management”:** This chapter delves into the commands that manipulate files. You create, delete, compress, move, and look at the files on disk and gain a solid understanding of how the Linux filesystems operate.
- **Chapter 6, “Text Processing/Advanced Command Line”:** The Linux command line is a programming environment that lets you do complicated tasks with a few keystrokes. This chapter introduce you to the most powerful feature of the shell of all: chaining together individual commands into increasingly powerful command lines. Along the way you learn how to search through text using regular expressions.
- **Chapter 7, “Process Management”:** Things that run on a Linux system are called processes, and this chapter teaches you how to manipulate these processes. You learn how to start and stop processes, run them in the background, and see which ones are taking the most resources from your computer.
- **Chapter 8, “Editing Text”:** This chapter teaches you to be productive in the vim editor. Vim makes repetitive tasks a breeze and lets you perform powerful edits on text files without moving the mouse. As most configuration and

programming on Linux is through a text file, an administrator who can wield a text editor with efficiency is one who has her work done on time.

- **Chapter 9, “Partitions and Filesystems”:** This chapter takes a deep dive into how a Linux system uses disks. You learn how filesystems work and how to add and remove capacity from a Linux workstation.
- **Chapter 10, “Permissions and Ownership”:** Linux was built as a multiuser system from the very beginning, so an understanding of how access to resources is granted and checked is important to maintain the security of your data and the sanity of your users. This chapter investigates the Linux permission model along with the commands used to check and set permissions.
- **Chapter 11, “Customizing Shell Environments”:** This chapter explores ways that you can customize your command line, such as by making shorter versions of longer commands or adding your own functions to the command line. Here, we also look at the roles played by internationalization and localization, which are methods that let the shell adapt to different languages and countries without needing to maintain multiple installations.
- **Chapter 12, “Shell Scripting”:** The Linux shell is actually a sophisticated programming environment and this chapter shows you the basics. You don’t have to be a programmer to write shell scripts—this chapter starts with the most basic script and works from there.
- **Chapter 13, “Basic SQL Management”:** The Structured Query Language is a way that databases query and manipulate data. This chapter, through real world examples, teaches you the basics of SQL so that you can more effectively help your users and answer questions about your own data.
- **Chapter 14, “Configuring User Interfaces and Desktops”:** Linux isn’t just a command line system—there are many graphical tools from word processors to video games. This chapter shows you how to use Linux in a graphical mode.
- **Chapter 15, “Managing Users and Groups”:** Users and groups are the other half of the Linux permissions model that was started in Chapter 10. This chapter teaches the administrative tasks associated with managing the users on your system.
- **Chapter 16, “Schedule and Automate Tasks”:** This chapter walks you through the various ways that Linux systems can run tasks without user intervention, such as to process statistics from logs while you’re sleeping.
- **Chapter 17, “Configuring Print and Email Services”:** This chapter looks at two basic services that Linux is often called to solve: printing and email. With printing, you learn how the Common Unix Printing System (CUPS) is put together and how it can be used to manage printing for a single system or

a large enterprise. In the email half of the chapter you learn how email works and what software is used on Linux to perform the various roles in an Internet email system. You also see how to do basic account management in an email system.

- **Chapter 18, “Logging and Time Services”:** Logs provide a detailed accounting of what happened when you weren’t looking. This chapter explains the Linux logging systems and how to configure and use them. Additionally you learn how time is kept on a Linux system and how different Linux systems can talk to coordinate their time.
- **Chapter 19, “Networking Fundamentals”:** A Linux system that provides network services is only as good as its network configuration. This chapter gives you the solid understanding of networking needed to determine whether Linux or the network is causing a problem. You also learn about the various services used to connect computers on a network.
- **Chapter 20, “System Security”:** Security is all about assessing the risk to your machine and keeping the bad guys out. In this chapter you learn how to assess the security of your system, lock down services to only people you want, and encrypt your data from prying eyes.
- **Chapter 21, “Final Preparation”:** In this final chapter you find exam questions that challenge your understanding of the material and provide a test that assesses your readiness to take either the LPIC 101 or Linux+ exams.
- **Glossary:** The glossary defines all terms that you were asked to define at the end of each chapter.

Each chapter follows the same format and incorporates the following tools to assist you by assessing your current knowledge and emphasizing specific areas of interest within the chapter:

- **“Do I Know This Already?” Quizzes:** Each chapter begins with a quiz to help you assess your current knowledge of the subject. The quiz is divided into specific areas of emphasis that enable you to best determine where to focus your efforts when working through the chapter.
- **Foundation Topics:** The foundation topics are the core sections of each chapter. They focus on the specific commands, concepts, or skills that you must master to successfully prepare for the examination.
- **Exam Preparation Tasks:** At the end of the foundation topics, the Exam Preparation Tasks highlight the key topics from the chapter and lists the pages where you can find them for quick review. This section also provides a list of key terms that you should be able to define in preparation for the exam. It is unlikely that you will be able to successfully complete the certification exam

by just studying the key topics and key terms, although they are a good tool for last-minute preparation just before taking the exam. For a thorough understanding of how to prepare for the exam, see Chapter 21.

- **Review Questions:** Questions at the end of each chapter measure your understanding of the topics discussed in the chapter.
- **DVD-Based Practice Exam:** This book includes a DVD containing several interactive practice exams. It is recommended that you continue to test your knowledge and test-taking skills by using these exams. You will find that your test-taking skills improve by continued exposure to the test format. Remember that the potential range of exam questions is limitless. Therefore, your goal should not be to “know” every possible answer but to have a sufficient understanding of the subject matter so that you can figure out the correct answer with the information provided.

Pearson IT Certification Practice Test Engine and Questions on the DVD

The DVD in the back of the book includes the Pearson IT Certification Practice Test engine—software that displays and grades a set of exam-realistic multiple-choice questions. Using the Pearson IT Certification Practice Test engine, you can either study by going through the questions in Study Mode, or take a simulated exam that mimics real exam conditions. You can also serve up questions in a Flash Card Mode, which displays just the question and no answers, challenging you to state the answer in your own words before checking the actual answers to verify your work.

The installation process requires two major steps: installing the software and then activating the exam. The DVD in the back of this book has a recent copy of the Pearson IT Certification Practice Test engine. The practice exam (the database of exam questions) is not on the DVD.

Note The cardboard DVD case in the back of this book includes the DVD and a piece of paper. The paper lists the activation code for the practice exam associated with this book. Do not lose the activation code. On the opposite side of the paper from the activation code is a unique, one-time-use coupon code for the purchase of the Premium Edition eBook and Practice Test.

Install the Software from the DVD

The Pearson IT Certification Practice Test is a Windows-only desktop application. Unfortunately, you cannot easily run this .exe on a Linux machine. You can run it on a Mac using a Windows virtual machine, but it was built specifically for the PC platform. The minimum system requirements are as follows:

- Windows 10, Windows 8.1, Windows 7, or Vista (SP2)
- Microsoft .NET Framework 4.0 Client
- Pentium-class 1 GHz processor (or equivalent)
- 512 MB RAM
- 650 MB disk space plus 50 MB for each downloaded practice exam
- Access to the Internet to register and download exam databases

The software installation process is routine as compared with other software installation processes. If you have already installed the Pearson IT Certification Practice Test software from another Pearson product, there is no need for you to reinstall the software. Simply launch the software on your desktop and proceed to activate the practice exam from this book by using the activation code included in the DVD sleeve.

The following steps outline the installation process:

1. Insert the DVD into your PC.
2. The media interface that automatically runs allows you to access and use all DVD-based features, including the exam engine and sample content from other Cisco self-study products. From the main menu, click the Install the Exam Engine option.
3. Respond to windows prompts as with any typical software installation process.

The installation process gives you the option to activate your exam with the activation code supplied on the paper in the DVD sleeve. This process requires that you establish a Pearson website login. You need this login to activate the exam, so please do register when prompted. If you already have a Pearson website login, there is no need to register again. Just use your existing login.

Activate and Download the Practice Exam

Once the exam engine is installed, you should then activate the exam associated with this book (if you did not do so during the installation process) as follows:

1. Start the Pearson IT Certification Practice Test software from the Windows Start menu or from your desktop shortcut icon.
2. To activate and download the exam associated with this book, from the My Products or Tools tab, click the Activate Exam button.
3. At the next screen, enter the activation key from the paper inside the cardboard DVD holder in the back of the book. Once entered, click the Activate button.
4. The activation process downloads the practice exam. Click Next and then click Finish.

When the activation process completes, the My Products tab should list your new exam. If you do not see the exam, make sure that you have selected the My Products tab on the menu. At this point, the software and practice exam are ready to use. Simply select the exam and click the Open Exam button.

To update a particular exam you have already activated and downloaded, display the Tools tab and click the Update Products button. Updating your exams ensures that you have the latest changes and updates to the exam data.

If you want to check for updates to the Pearson Cert Practice Test exam engine software, display the Tools tab and click the Update Application button. You can then ensure that you are running the latest version of the software engine.

Activating Other Exams

The exam software installation process and the registration process, only have to happen once. Then, for each new exam, only a few steps are required. For instance, if you buy another Pearson IT Certification Cert Guide, extract the activation code from the DVD sleeve in the back of that book; you do not even need the DVD at this point. From there, all you have to do is start the exam engine (if not still up and running) and perform steps 2 through 4 from the previous list.

Certification Exam Topics and This Book

The questions for each certification exam are a closely guarded secret. However, we do know which topics you must know to successfully complete this exam. CompTIA and LPI publishes them as an exam blueprint.

Tables I.1 and I.2 list the exam topics for each exam.

Table I.1 CompTIA Linux+ (LX0-103) and LPIC-1 (101-400) Exam

Exam Topics for CompTIA Linux+ (LX0-103) and LPIC-1 (101-400) Exam
Topic 101: System Architecture
101.1 Determine and configure hardware settings
101.2 Boot the system
101.3 Change run levels/boot targets and shutdown or reboot system
Topic 102: Linux Installation and Package Management
102.1 Design hard disk layout
102.2 Install a boot manager
102.3 Manage shared libraries
102.4 Use Debian package management
102.5 Use RPM and YUM package management
Topic 103: GNU and Unix Commands
103.1 Work on the command line
103.2 Process text streams using filters
103.3 Perform basic file management
103.4 Use streams, pipes, and redirects
103.5 Create, monitor, and kill processes
103.6 Modify process execution priorities
103.7 Search text files using regular expressions
103.8 Perform basic file editing operations using vi
Topic 104: Devices, Linux Filesystems, Filesystem Hierarchy Standard
104.1 Create partitions and filesystems
104.2 Maintain the integrity of filesystems
104.3 Control mounting and unmounting of filesystems
104.4 Manage disk quotas
104.5 Manage file permissions and ownership
104.6 Create and change hard and symbolic links
104.7 Find system files and place files in the correct location

Table I-2 CompTIA Linux+ (LX0-104) and LPIC-1 (102-400)

Exam Topics for CompTIA Linux+ (LX0-104) and LPIC-1 (102-400)
Topic 105: Shells, Scripting, and Data Management
105.1 Customize and use the shell environment
105.2 Customize or write simple scripts
105.3 SQL data management
Topic 106: User Interfaces and Desktops
106.1 Install and configure X11
106.2 Set up a display manager
106.3 Accessibility
Topic 107: Administrative Tasks
107.1 Manage user and group accounts and related system files
107.2 Automate system administration tasks by scheduling jobs
107.3 Localization and internationalization
Topic 108: Essential System Services
108.1 Maintain system time
108.2 System logging
108.3 Mail Transfer Agent (MTA) basics
108.4 Manage printers and printing
Topic 109: Networking Fundamentals
109.1 Fundamentals of Internet protocols
109.2 Basic network configuration
109.3 Basic network troubleshooting
109.4 Configure client side DNS
Topic 110: Security
110.1 Perform security administration tasks
110.2 Set up host security
110.3 Securing data with encryption

Assessing Exam Readiness

Exam candidates never really know whether they are adequately prepared for the exam until they have completed about 30% of the questions. At that point, if you are not prepared, it is too late. The best way to determine your readiness is to work through the “Do I Know This Already?” quizzes at the beginning of each chapter and review the foundation and key topics presented in each chapter. It is best to work your way through the entire book unless you can complete each subject without having to do any research or look up any answers.

Exam Registration

For LPI exams, start at lpi.org to get a member ID and a link to pearsonvue.com/lpi/ to schedule an exam. For the Linux+ variants, sign up directly from <https://certification.comptia.org/certifications/linux>.

Where Are the Companion Content Files?

Register this print version of *CompTIA Linux+ / LPIC-1 Cert Guide* to access the content from the DVD online.

This print version of this title comes with a disc of companion content. You have online access to these files by following these steps:

1. Go to www.pearsonITcertification.com/register and log in or create a new account.
2. Enter the ISBN: 9780789754554.
3. Answer the challenge question as proof of purchase.
4. Click on the Access Bonus Content link in the Registered Products section of your account page to be taken to the page where your downloadable content is available.

Please note that many of our companion content files can be very large, especially image and video files.

If you are unable to locate the files for this title by following these steps, please visit www.pearsonITcertification.com/ contact and select the Site Problems/Comments option. Our customer service representatives will assist you.

This page intentionally left blank



This chapter covers the following topics:

- Filesystem Overview
- File Management Commands
- Where Are Those Files?
- Backup Commands

This chapter covers the following objectives:

- Perform basic file management: 103.3
- Create and change hard and symbolic links: 104.6
- Find system files and place files in the correct location: 104.7

File Management

Most of what you do on a Linux machine involves manipulating files in some manner. You have to know where certain files go, such as binaries, configuration, and user data. You also need to be able to manipulate files from the command line rather than a GUI.

“Do I Know This Already?” Quiz

The “Do I Know This Already?” quiz enables you to assess whether you should read this entire chapter or simply jump to the “Exam Preparation Tasks” section for review. If you are in doubt, read the entire chapter. Table 5-1 outlines the major headings in this chapter and the corresponding “Do I Know This Already?” quiz questions. You can find the answers in Appendix A, “Answers to the ‘Do I Know This Already?’ Quizzes and Review Questions.”

Table 5-1 “Do I Know This Already?” Foundation Topics Section-to-Question Mapping

Foundation Topics Section	Questions Covered in This Section
Filesystem Overview	1, 3
File Management Commands	2, 4-6
Where Are Those Files?	7-8
Backup Commands	9-11

1. Files that change often should go under:
 - a. /usr
 - b. /proc
 - c. /root
 - d. /var

2. Your shell is in `/usr/local`. You type `cd ../bin`. Which directory is shown when you type `pwd`?
 - a. `/bin`
 - b. `/usr/bin`
 - c. `/usr/local/bin`
 - d. Nothing, this command returns an error.

3. Which of the following directories should be on the same partition as the root?
 - a. `/boot`
 - b. `/usr`
 - c. `/home`
 - d. `/sbin`

4. You happen across a file in a directory called `foo`. What is a good way to find out what the file is or does?
 - a. `file foo`
 - b. `/foo`
 - c. `cat foo`
 - d. `which foo`

5. What command would be used to update the date on a file?
 - a. `tar`
 - b. `file`
 - c. `date`
 - d. `touch`

6. You are trying to create a new series of nested directories: `/a/b/c/d/`. What is the fastest way to create this nested directory set?
 - a. `mkdir /a; mkdir /a/b; mkdir /a/b/c; mkdir /a/b/c/d`
 - b. `mkdir /a/b/c/d`
 - c. `mkdir -p /a/b/c/d`
 - d. `md /a/b/c/d`

7. You know that you have multiple copies of the **doit** command on your system. How do you find which one you will run if you type **doit** at the command line?
 - a. **whereis doit**
 - b. **locate doit**
 - c. **find doit**
 - d. **which doit**

8. You know that you downloaded a file called `backup.tar.gz` this morning but can't remember where you put it. Which is the most appropriate command to find the file?
 - a. **find / -name backup.tar.gz**
 - b. **find backup.tar.gz**
 - c. **locate backup.tar.gz**
 - d. **whereis backup.tar.gz**

9. You want to package up Fred's home directory on a USB stick to send with him as he's leaving your company. Which command is the best? Hurry, because there's cake!
 - a. **find /home/fred | tar -czf > /media/removable/fred.tar.gz**
 - b. **tar -czf /home/fred > /media/removable/fred.tar.gz**
 - c. **cd /home/; tar -cjf /media/removable/fred.tar.bz2 fred**
 - d. **cd /home/fred tar -cjf /media/removable/fred.tar.bz2 ***

10. What does the command **tar -tf archive.tar etc/pine.conf** do?
 - a. Makes a new archive called `archive.tar` containing `/etc/pine.conf`
 - b. Adds `etc/pine.conf` to `archive.tar`
 - c. Checks to see whether `etc/pine.conf` is inside the archive
 - d. Extracts `etc/pine.conf` from `archive.tar`

11. Which compression utility offers the highest level of compression?
 - a. **bzip2**
 - b. **gzip**
 - c. **compress**
 - d. **cpio**

Foundation Topics

Filesystem Overview

The filesystem's structure starts with the root of the filesystem, which is denoted by the forward slash character (/). Every item on the filesystem is accessible by a single unique path from the root of the system, such as /usr/local/bin/foobar, no matter which device that file is stored on.

Unix evolved its own set of traditions as to where certain files would go. The fragmentation of the commercial and academic Unixes led to differences in conventions depending on which flavor of Unix you were using.

Linux borrows practices from many different Unixes and has fragmentation of its own in the form of different distributions. The community started working on a standard for filesystem layout called the *File System Hierarchy Standard (FHS)* to make it easier for both people and software to know where files can be found.

The latest FHS is always found at <http://www.pathname.com/fhs/>.

LPI bases the exam questions about the directory structure from the FHS 2.3. The FHS isn't really a standard but a firm set of suggestions that most, but not all, distribution vendors obey. A good number of questions on the exams reference the FHS.

What Belongs Where

The exams make somewhat of a big deal about what the proper directories and locations are for Linux files, but few things are more vexing than to be asked what should positively be in the root (/) directory, or what can be elsewhere.

The Root of the System

Starting in the root (/) directory, the Table 5-2 lists common top-level directories and includes a short explanation for each:

Table 5-2 Common Directories

Directory	Description
bin	Binaries for all users
boot	Kernel, system map, boot files
dev	Device files
etc	Configuration files for the host

Directory	Description
home	Home directories for users
lib	Necessary shared libraries/modules
lost+found	Storage directory for unlinked files (found with fsck)
media	Mount points for removable media
mnt	Temporary mount point for the sysadmin
opt	Third-party application software
proc	Kernel and process information
root	The root user's home directory
sbin	System binaries needed for boot
tmp	Temporary data
usr	Sharable, read-only data and programs, no host-specific data
var	Variable data, logs, Web, FTP, and so on

The exam makes a big deal out of what's optional and required in the **root (/)** directory. If you read the FHS 2.3 (highly recommended), you see that the word “optional” appears next to the **/root** and **/home** directories. It is possible that the computer is some kind of application server where users are not expected to log in. This is key because you'll be asked questions about which directories are optional in the root filesystem.

Key Topic

The FHS documentation states, “The contents of the root filesystem must be adequate to boot, restore, recover, and/or repair the system. To boot a system, enough must be present on the root partition to mount other filesystems. This includes utilities, configuration, boot loader information, and other essential start-up data. /usr, /opt, and /var are designed such that they may be located on other partitions or filesystems.”

From this statement you can understand which of the preceding directories need to be on the root partition and which can be moved to other partitions.

Classifying Data

FHS makes distinctions between data that changes and data that is static, and data that can be shared versus data that's local to the computer. Data of different categories should be separated into different directories.

Because of the way the FHS is laid out, with the root filesystem being described in section 3 and **/usr** and **/var** happening later, it's easy to misunderstand what is really

supposed to be on the root filesystem as opposed to another device that's mounted after boot.

The relationship between **/usr** and **/var** is that, long ago in Unix times, **/usr** used to contain all types of data. The FHS tried to extract the data that changes and is non-sharable to **/var**, leaving **/usr** with just static, sharable data.

Where Programs Live

The FHS does not allow programs to create their individual named directories in the **/usr** section. The subdirectories allowed to exist directly under the **/usr** directory are

- **bin**—Contains user commands
- **include**—Contains header files for C programs
- **lib**—Contains libraries
- **local**—Contains local/sharable programs
- **sbin**—Contains nonessential system binaries
- **share**—Contains data/programs for multiple architectures

Key Topic

The **/usr** section has a location for programs named **/usr/local**. This is for the sys-admin to install software in a place that won't conflict with the distribution files. Programs in the **/usr/local** path are also allowed for sharing among groups of hosts.

For example, say your developers have come up with a program to calculate loans and you want to install it on the workgroup server for other systems to remotely mount and use. Because this is a third-party or custom application, the logical place for it is in **/usr/local/appname**, possibly with a link to the program binary in the **/usr/local/bin** directory (because that's where local binaries are expected to be found).

If given a choice between putting the software package **BIGPROG** in the **/usr/local/BIGPROG** section and the **/opt/BIGPROG** section, it's hard to choose. Read any relevant exam question closely—the main difference being that the **/opt** section is not considered to be sharable, whereas the **/usr** section is often shared and mounted by client systems.

File Management Commands

A major section of the 101 exam is dedicated to how to run commands properly with the right options and arguments. As a good sysadmin, you are expected to know how to create, delete, edit, set permissions, display, move, copy, and determine the type of files and programs.

Tips for Working with Linux Files

Because most users and sysadmins come from a Windows or other OS background, a quick set of recommendations for the less-experienced can be of help here:

- **Hidden files aren't really hidden**—They just begin with a `.`, such as the `.bashrc` and `.bash_profile` files. They are normally not visible unless you explicitly ask for them to be displayed and aren't deleted by commands such as `rm -f *.*`.
- **Filenames can contain multiple periods or no period characters**—The filenames `this.is.a.long.file` and `thisisalongfile` are perfectly reasonable and possible.
- **Spaces in filenames look nice, but are a pain to type**—Use an `_` or a `-` instead of spaces because it's neater and easier than prefixing all spaces with a `\`. (To display a space in a filename, the system shows a space prefixed with a backslash.)
- **File extensions aren't mandatory**—But they are useful for sorting, selection, and copy/move/delete commands, as well as for quickly identifying a file's type.

Basic Navigation

The command to change the current working directory, `cd`, is used frequently and knowing how to move around the filesystem is a main focus of the exams.

The following command simply moves you from wherever you are to the `/etc` directory. This type of move uses absolute pathnames and can be used from within any directory:

```
cd /etc
```

The path is called *absolute* because it defines a path starting at the root of the filesystem. The easy way to tell whether the path is absolute is that it starts with a slash (`/`).

Moving relatively from the current directory to a subdirectory is quick and easy, such as if you are in the `/etc/` directory and want to change into the `/etc/samba` directory. Here's how:

```
cd samba
```

This is referred to as a *relative path* because the option you pass to the **cd** command is relative to the current directory. You are in **/etc** and moving to **samba** gets you in **/etc/samba**. If you were in **/home** and ran **cd samba** it would not work unless **/home/samba** also existed.

If you get confused as to where you currently are, use the **pwd** command to print the working (current) directory:

```
# pwd
/etc/samba
```

By itself, the **cd** command takes you back to your home directory, wherever you happen to be. The tilde (~) also means “home directory,” so **cd ~** takes you to your home directory and **cd ~sean** takes you to Sean’s home directory.

Advanced Navigation

It’s good to get experience with some complex relative path situations. For example, if you were in the directory **/home1/user1** and wanted to move into the directory **/home2/user2**, which command could be used?

```
$ tree /
/
|-- home1
|   |-- user1
|-- home2
|   |-- user2
```

Remember, you aren’t using absolute pathnames, just relative pathnames.

The answer is

```
# cd ../../home2/user2
```

Key Topic

Each of the **..** pairs takes you up one level: The first takes you to **/home1** and the second puts you at the root. From there it’s relative pathnames. Practice this method, and remember that going up one level in this exercise only got you to the **/home1** directory. This is a relative path because the path does not start with a **/**. The directory in which you end up depends on where you started.

Though this example of relative and absolute pathnames was used to look at changing directories, it applies to any situation where you’re prompted for a filename.

Listing Files and Directories

The **ls** command is used for listing directories or files, or both.

If you use the **ls** command to see a multicolumn output of the current directory, only the file or directory names are shown, not other details about the file:

```
ls
file1 file2 file3 file4
```

Use the **-l** long listing option to see all the details of a particular file or directory, or set of files or directories in a single column, like so:

```
$ ls -l
total 0
-rw-r--r--  1 root   root           0 Jan 24 18:55 file1
-rw-r--r--  1 root   root           0 Jan 24 18:55 file2
-rw-r--r--  1 root   root           0 Jan 24 18:55 file3
-rw-r--r--  1 root   root           0 Jan 24 18:55 file4
```

Key Topic

The **-l** long listing style is the only way to use the **ls** command and see the permissions, ownership, and link counts for objects. The only other command that can give such information is the **stat** command, which shows a single filesystem object at a time.

Other examples of using the **ls** command include

- **ls /home/user**—Shows a plain listing of that directory.
- **ls -a**—Lists all files, including hidden `.` files.
- **ls -d foo**—Lists just the directory called `foo`, not the contents.
- **ls -i**—Lists the inode number for the targetfile or directory. Inodes are the way Linux represents a file on disk and are discussed later in the section “Copying Files and Directories.”
- **ls -l**—Shows permissions; links; and date, group, and owner information. Permissions dictate who can access the file and are discussed in detail in Chapter 10, “Permissions.”
- **ls -lh**—Shows human-readable output of file sizes, in KB, MB, and GB, along with file details.

Chaining the options together produces useful results. For example, if you needed to see all the files (including hidden ones) in the current directory, their permissions, and their inode numbers, you would use the following command:

```
# ls -lai
290305 drwxr-x--- 13 root   root           4096 Jan 24 18:55 .
          2 drwxr-xr-x 20 root   root           4096 Jan 24 17:56 ..
292606 -rw-r--r--  1 root   root           1354 Jan 21 00:23 anaconda-ks.
cfg
```

```

292748 -rw-----  1 root  root    3470 Jan 24 18:16 .bash_history
290485 -rw-r--r--   1 root  root     24 Jun 10  2000 .bash_logout
290486 -rw-r--r--   1 root  root    234 Jul  5  2001 .bash_profile
290487 -rw-r--r--   1 root  root    176 Aug 23  1995 .bashrc
290488 -rw-r--r--   1 root  root    210 Jun 10  2000 .cshrc

```

Determining File Types

With no requirement for extensions on Linux files, a tool for easily determining file types is essential. The **file** command can be used to read the file’s headers and match that data against a known set of types.

The **file** command uses several possible sources, including the **stat** system call, the magic number file (**/usr/share/magic**), and a table of character sets including ASCII and EBCDIC. Finally, if the file is text and contains recognizable strings from a given programming or other language, it is used to identify the file.

The output can be used, manipulated, and filtered to show you useful things.

For example, simply using the **file** command on a given file shows the type:

```

$ file file1
file1: ASCII text

```

Running the **file** command against a known binary shows various elements about the architecture and layout of the file, such as shown here:

```

$ file /bin/ls
/bin/ls: ELF 32-bit LSB executable, Intel 80386, version 1 (SYSV),
dynamically linked (uses shared libs), for GNU/Linux 2.6.32, stripped

```

Running the **file** command against a directory full of files is useful for viewing the possible types, but the real gold lies in filtering the output using the pipe operator (**|**) and the **grep** command, showing only the results that contain the word “empty”:

```

$ file /etc/* | grep empty
/etc/dumpdates:          empty
/etc/exports:           empty
/etc/fstab.REVOKE:     empty
/etc/motd:              empty
/etc/printconf.local:  empty

```

This is one way of finding empty files that are littering your system. They are probably required in the **/etc** directory but only clutter temporary directories such as **/tmp**.

NOTE The asterisk (*) in the previous command is known as a glob. A *glob* is a wildcard operator that matches some number of files based on a pattern. `/etc/*` matches all files in the `/etc` directory such as `/etc/foo`, `/etc/bar`, but not `/etc/foo/bar`!

One thing that's distinct about Linux (and all Unixes) is that the shell is responsible for expanding the glob to the list of files it matches. If you type `ls /tmp/thing*` and there are two files that start with thing such as **thing1** and **thing2**, it's the same thing as if you typed `ls /tmp/thing1 /tmp/thing2`:

```
$ ls thing*
thing1  thing2
```

This globbing feature is why renaming a group of files is harder. In Windows you could type `ren *.foo *.bar` and any file with an extension of `foo` would then have an extension of `bar`. In Linux, typing `mv *.foo *.bar` would expand the globs to the list of files matched—`*.foo` would match the files you want to rename and `*.bar` would match nothing. This is different from what you might expect! The following output shows this problem.

```
$ ls *.foo *.bar
ls: *.bar: No such file or directory
file1.foo  file2.foo
$ echo mv *.foo *.bar
mv file1.foo file2.foo *.bar
$ mv *.foo *.bar
mv: target `*.bar' is not a directory
```

In the output, the first command shows there are three files with an extension of **foo** but none of **bar**. The **echo** command displays the output that follows it, such that it shows what would be executed if you ran the **mv** command by itself. The `*.bar` glob shows up because there are no files that match it. The error happens because there is no such directory called `*.bar`.

There are other glob operators. Example 5-1 shows some uses of file globs.

Example 5-1 Examples Using a Glob

```
$ ls
file file1 file10 file11 file2
$ ls file*
file file1 file10 file11 file2
$ ls file?
file1 file2
```



```
$ ls *1
file1 file11
$ ls file[123]
file1 file2
```

Example 5-1 starts by listing all the files in the directory. The same list of files is also available with **file***, which matches the word “file” followed by anything, or nothing at all. Note how it includes the bare name “file”. Next the **file?** glob matches anything starting with the word “file” and followed by one character. Both “file” and the files with two-digit numbers in their names are excluded.

Globs don’t have to appear at the end of a filename. ***1** matches anything ending in the number “1”. Finally, **file[123]** uses the square bracket operator that means “any one character from the set”. This matches file1 and file2.

Touching Files

The **touch** command seems odd at first, but it comes in handy often. You give it the name of one or more files, and it creates the files if they don’t exist or updates their timestamps if they do.

There are various reasons to use the **touch** command, such as creating a new blank log file or updating a file’s modification time to use as a reference such as to know the last time a job was run.

To create a new file, you can use the relative pathname for creating one in the current directory:

```
touch filename
```

Or, you can use absolute pathname to create the file, such as shown here:

```
touch /home/rossb/filename
```

Key Topic

Expect to see **touch** on the exams for log file creation, along with using a reference file to mark the last backup. In other words, if a log file is created from a successful backup, that file can be used as a date and time reference file because it occurred at a desirable time.

When you use **touch** on an existing file, the default action is to update all three of the file’s times:

- **access**—The last time a file was written/read from

- **change**—The last time the contents of the file were changed, or that the file’s metadata (owner, permission, inode number) was changed
- **modify**—The last time the file’s contents were changed

A programmer preparing a new release of a software package would use the **touch** command to ensure that all files have the exact same date and times. Therefore, the release could be referred to by the file date, given multiple revisions.

Setting a file’s date is relatively easy; the following command sets **file1**’s date to a particular date and time:

```
touch -t 201501010830 file1
```

The time format used is represented by `yyyymmddhhmm`, or a four-digit year, two-digit month, two-digit day, two-digit hour, and two-digit minutes.

Reference files are useful, particularly when you just want to have a file or set of files updated to a particular date/time, not the current one. You could use

```
touch -r reffile file2update
```

The date and time of **reffile** is applied to the **file2update** file date and time.

Copying Files and Directories

One aspect of copying an object is that the act creates a new file with a separate inode. This means that the operating system sees the new file as separate from the old one. Contrast this to a move operation where it’s the same file with a new name.

When you create an object in a filesystem, it gets its own permissions. **cp** doesn’t always copy the permissions over to the new file. This can be done, but it requires the use of the **-p** option to preserve the permissions and ownership. The root user is the only user that can change the ownership of a file; therefore, regular users using this option always own the copied files no matter who the original owner was.

A normal copy is simple to perform. You’re essentially causing the file to be replicated to the new location:

```
cp file1 /dir1/file2
```

A few options that make life easier for copying files include

- **-d**—Doesn’t follow symbolic links; copies the link instead. Links point one file to another and are explored later in the “Linking Files” section.
- **-f**—Force overwriting existing files.
- **-i**—Interactively asks before overwriting.

- **-l**—Creates a hard link to the source file.
- **-r** or **-R**—Recursively traverses directories (copying everything).
- **-s**—Creates a symlink to the source file.
- **-u**—Only updates the copy when the source is newer than the target or the target doesn't exist.
- **-x**—Doesn't traverse to filesystems mounted from other devices.

Copying an existing directory to a new one is simple:

```
# cp -r dir1 dir2
```

The **-r** option is necessary because the **cp** command doesn't process directories by default. As long as the target directory does not exist, the previous command makes an identical copy of the source and all subordinate files and directories in the target directory.

Copying a source directory to an existing target directory doesn't attempt an overwrite; it makes the source directory into a new subdirectory of the target.

For example, if you are in the **/test** directory and have the structure shown in the following, you might assume that issuing a **cp -r dir1 dir2** would overwrite **dir2**, or at least prompt you to see whether you wanted to:

```
$ tree .
|-- dir1
|   |-- file1
|   `-- subdir1
`-- dir2
```

When you issue the **cp -r dir1 dir2** command, the filesystem (along with the **cp** command) notices the existing **dir2** entry and automatically drops the source directory into **dir2** as a subdirectory, like this:

```
-- dir1
|   |-- file1
|   `-- subdir1
`-- dir2
    |-- dir1
        |-- file1
        `-- subdir1
```

Key Topic

The correct way to copy the contents of **dir1** into **dir2**, thereby mirroring **dir1** exactly, is to focus on the word “contents.” By suffixing the source (**dir1**) with a forward slash and an asterisk (**dir1/***), you tell the **cp** command to ignore the directory entry and focus on the filenames inside the directory.

With the same initial setup, if you issue the command `cp -r dir1/* dir2`, you get the correct results:

```
$ tree .
|-- dir1
|   |-- file1
|   `-- subdir1
`-- dir2
    |-- file1
    `-- subdir1
```

The inability to properly copy a directory or its contents will come back to haunt you on the exam. In addition, if you see a source directory with only a trailing forward slash (`dir1/`) but no asterisk, it's identical to using (`dir1`). In other words, to copy just the contents of a directory, you have to address them specifically with the forward slash and asterisk (`dir1/*`).

Two special characters used in relative directory naming are often used when copying files. The current directory is represented by a single period (`.`) and the parent directory by two periods (`..`).

For example, if you are currently in the `/home/rossb` directory and want to copy a set of files from the `/home/lukec` directory, you can avoid typing the full path of the current directory with the (`.`) character. Both of these commands perform the same action:

```
cp /home/lukec/*.mp3 .
cp /home/lukec/*.mp3 /home/rossb
```

Moving Objects

Where the `cp` command copies a file by creating a new file, inode, and data, the `mv` command simply changes which directory file contains the file or directory entry or alters the entry in the file if it stays in the same directory. By changing just the metadata that points to the file, moving a file on the same device is quick. If the file move happens across two devices, the file is copied to the new device and deleted from the old one.

Create a file named `file1`; then run the `stat` command on it to check the details, as shown in Example 5-2.

Example 5-2 Running the **stat** Command on **file1**

```

$ touch file1
$ stat file1
  File: `file1'
  Size: 0          Blocks: 0          IO Block: 4096   regular empty
file
Device: fd00h/64768d      Inode: 2261179   Links: 1
Access: (0664/-rw-rw-r--) Uid: (500/sean)   Gid: (500/sean)
Access: 2015-02-03 21:47:46.000000000 -0600
Modify: 2015-02-03 21:47:46.000000000 -0600
Change: 2015-02-03 21:47:46.000000000 -0600
Birth: -

```

Now move the file to a new name with the **mv** command, as shown in Example 5-3.

Example 5-3 Moving Files to a New Name

```

$ mv file1 file2
$ stat file2
  File: `file2'
  Size: 0          Blocks: 0          IO Block: 4096   regular empty
file
Device: fd00h/64768d      Inode: 2261179   Links: 1
Access: (0664/-rw-rw-r--) Uid: (500/sean)   Gid: (500/sean)
Access: 2015-02-03 21:47:46.000000000 -0600
Modify: 2015-02-03 21:47:46.000000000 -0600
Change: 2015-02-03 21:48:41.000000000 -0600
Birth: -

```

Because the device and inode stayed the same you know this is the same file as before. The change time was modified to reflect the fact that the file was renamed.

When you move a file, the **mv** command overwrites the destination if it exists. This command supports an option, **-i**, that first checks the target to see whether it exists. If it does, **mv** asks whether you want to overwrite the target. Some distributions make **-i** a default option with a shell alias. Chapter 11, “Customizing Shell Environments,” discusses shell aliases in more detail.

Another quirk of the command is the lack of an **-r**, or recursive, option. This is because when you move a directory or a file you’re just changing the directory entry for the file. The directory continues to point to the same files so there is no need to move the files themselves.

You can avoid the overwriting of newer target files or directories with the **-u** option, preserving the latest copy of an object.

Examples of moving files and directories include moving a single directory to another directory name, as shown here:

```
mv -f dir1 dir2
```

This merely changes the directory entry **dir1** to the new name **dir2**. It also removes the “are-you-sure” prompt with the **-f** option.

Just like the **cp** command, moving directory contents requires a correctly formed command; otherwise, you’ll move a directory not to the new name, but to a subdirectory of the existing directory.

For example, consider the **/test** directory again, with its structure similar to the following:

```
$ tree .
|-- dir1
|   |-- file1
|   `-- subdir1
`-- dir2
```

If you were a Windows administrator, it would make sense to run the following command to move **dir1** to **dir2**:

```
mv dir1 dir2
```

If you do this on a Linux system and then run the **tree** command, you see the following output:

```
$ tree .
`-- dir2
    |-- dir1
        |-- file1
        `-- subdir1
```

This moves **dir1** under **dir2** because **dir2** already existed. To properly move the contents of the source **dir1** to the target **dir2**, you don’t need to use the nonexistent **-r** option (exam trick). You can just use a forward slash and an asterisk to refer to the files underneath **dir1**, like this:

```
mv dir1/* dir2
```

NOTE The `*` wildcard operator won't match hidden files because they begin with a period. Handling this case is actually quite complicated and outside the scope of the exam.

If you run the **tree** command, you see the following output:

```
$ tree .
|-- dir1
^-- dir2
    |-- file1
    ^-- subdir1
```

Finally, the directories you pass to the **mv** command don't always have to be underneath your current directory. You can use absolute pathnames, such as **mv /dir1 .** to move **dir1**, which is off the root directory into the current directory. You can also run **mv /dir1 /tmp** from anywhere in the system to move that same directory into the temporary directory.

Transforming Data Formats

The **dd** command is useful for a variety of tasks, not the least of which is creating backup images, called ISO files, of CD or DVDs. The two main formats **dd** interacts with are the raw device file and the full path of a file or object on the system.

For example, when creating a new boot disk, the **.img** binary file is read block by block from the CD-ROM (as a file) and written to a USB disk raw device as a set of blocks:

```
dd if=/mnt/cdrom/images/boot.img of=/dev/sdb
```

Creating an image of a CD-ROM involves reading the raw USB device block by block and creating a file on the filesystem that contains all those blocks:

```
dd if=/dev/sdb of=/root/usb.img
```

To duplicate a USB device named **sdb** to another USB device named **sdC**, the command is

```
dd if=/dev/sdb of=/dev/sdc
```

The **if** keyword means input file and the **of** keyword means output file. The exact order is unimportant, but as you can imagine, mixing up the in and out files can cause you to do terrible things such as overwriting parts of your hard drive!

dd, unlike most other Unix utilities, does not use dashes for its options. Options are specified in the format of **option=value**.

The **dd** command is also often used to duplicate a drive or partition of a drive to another like object.

For example, to copy the first partition from the `/dev/sda` disk to the same location on the second hard drive on the system, you would use the following command:

```
dd if=/dev/sda1 of=/dev/sdb1
```

You can also copy an entire disk device to another on the system by leaving off the partition numbers:

```
dd if=/dev/sda of=/dev/sdb
```

This works only if the second device is as large as or larger than the first; otherwise, you get truncated and worthless partitions on the second one.

Backing up the MBR is another trick that **dd** does well. Remember that the master boot record contains the indexes of all the partitions on that drive, and thus is very important. To create a disk file that contains only the first 512 bytes of the first hard drive in the system, use this command:

```
dd if=/dev/sda of=/root/MBR.img count=1 bs=512
```

The **count** keyword sets the number of reads from the input file you want to retrieve, and the **bs** keyword sets the block size.

If you don't set the count and block size on this command to back up the MBR, you'll be copying the entire device's blocks to the filesystem—a snake-eating-its-own-tail operation that is guaranteed to fill up the partition quickly and crash the system.

The restoration procedure is just the opposite:

```
dd if=/root/MBR.img of=/dev/sda count=1 bs=512
```

Creating and Removing Directories

A basic task of file management is to be able to create and remove directories, sometimes creating or removing whole trees at once. To create a directory named **dir1**, you use **mkdir dir1**. To create a directory named **subdir1** in the **dir1** directory, you use **mkdir dir1/subdir1**.

Always think of the last segment of any directory path as the object being created or removed, and think of the rest as supporting or parent objects. The **mkdir** and **rm-dir** commands are similar in features and options, including the capability of **mkdir** to create a deep subdirectory tree from scratch in a single command:

```
mkdir -p /dir1/dir2/dir3/dir4
```


One of the quirks about the **rmdir** command is that it cannot remove anything but an empty directory. For example, the last directory of the chain **/dir1/dir2/dir3/dir4** is the real target for this command, and only if that directory is empty (no regular or directory files) can it be removed.

```
rmdir -p /dir1/dir2/dir3/dir4
```

One option to the **rmdir** command does allow it to remove directories that have files and so on in them. It's called **--ignore-fail-on-non-empty** and is the longest option I know of in Linux. I'd rather type **rm -rf targetdir** 20 times than this beast.

Removing Objects

It follows that you'll want to remove objects after creating or copying them, and this is done with the **rm** command for most objects. **rmdir** can also be used.

Deleting files with the **rm** command is a matter of choosing the target to be removed and the options that work best.

If you want to remove a particular file and never be prompted by confirmation messages, the command is **rm -f target**.

To remove a directory and all its contents, and never get a confirmation message, the command is **rm -rf /full/path/to/target**.

Where Are Those Files?

Having a mechanism for finding or locating files on a Linux system is essential because the sheer amount of directories and files makes searching manually nearly impossible.

There are two methods for accomplishing this task—quick and dirty or slow and methodical. Most people try the quick **locate** command before resorting to the plodding **find** command.

Locating Files with Locate

The quickest way to find a file or set of files is to use the **locate** command. It's fast, database-driven, and secure. When you run the **locate** command you are searching a database instead of the filesystem, and only files that you have access to are shown. The downside of the database is that it's updated nightly and is therefore unaware of any changes that have happened since the last update.

locate has a quirky way of showing results. You would probably expect that using **locate** for a file named **readme** would locate only files named **readme**, but that's

not quite true. It finds anything that has a filename of **readme**, including regular files and any part of the path.

For example, while attempting to locate the **readme** file, you run the following command:

```
locate readme
```

This finds both of the following entries, one with the string **readme** as a part of the filename and the other a directory:

```
/readme
/usr/src/linux-2.4.20-8/drivers/net/wan/8253x/readme.txt
```

Key Topic

Use the **locate** command to find items you know are on the disk, or that you know existed before the last **locate** database update. The database that **locate** uses is updated nightly when the system runs its maintenance routines, or on demand. If you don't have permissions to the object, it isn't shown in the **locate** output.

Use **locate** with the **-i** option to ignore the case (upper or lower) and return anything that matches your search string using a case-insensitive match:

```
locate -i string
```

The **locate** database needs to be updated regularly to ensure good results. Your distribution probably puts it in the list of nightly jobs to be run. For more details on the nightly jobs, see Chapter 16, "Schedule and Automate Tasks." Updating the database can take a long time, and it is frustrating having to wait for the updates to finish when you need to search.

The **update** commands must be run as **root**, and either one will do the job:

```
updatedb
```

Sometimes you want to exclude files or directories from the **locate** database because they either are inappropriate or simply take too long to index without any apparent benefit. This is configurable in the **/etc/updatedb.conf** file. This file is read and the variables are used by the updating commands.

The two main methods of excluding objects in the configuration file are either by filesystem type or path. The following output is an example of a working **/etc/updatedb.conf** file:

```
PRUNEFS="devpts NFS nfs afs sfs proc smbfs autofs auto iso9660"
PRUNEPATHS="/tmp /usr/tmp /var/tmp /afs /net /sfs"
export PRUNEFS
export PRUNEPATHS
```

The **PRUNEFS** keyword is for filesystem types you want excluded from the **locate** database update; as you might expect, the **PRUNEPATHS** keyword is for directory trees you want excluded. Notice that most of the paths are temporary data locations or exotic file locations.

Key Topic

Remember for the exam that **locate** returns results for the search string in any portion of the path or filename it finds the string in. There will be questions that **locate** is right for, and some that really want the **whereis** command.

Finding Files

The **find** command is the most accurate but time-consuming method for searching the system for file objects because it crawls the list of files in real time versus the **locate** indexed database. The command consists of several (sometimes confusing) sections. But, if it's learned properly, it can be a powerhouse for the busy sysadmin.

The structure of a **find** command is

```
find startpath -options arguments
```

To make sense of this jumble of sections, let's take a look at a useful **find** command and match up the sections:

```
# find /home -iname *.mp3
/home/snuffly/g3 - red house.mp3
```

The previous command sets the start path to the **/home** directory and then looks for any instance of the string **mp3** as a file extension, or after the last **.** in the filename. It finds a file in the user **snuffly**'s home directory and returns the full path for that file.

Options for **find** include

- **group**—Based on files belonging to the specified group
- **newer**—Based on files more recent than the specified file
- **name**—Based on files with names matching a case-sensitive string
- **iname**—Based on files with names matching a non-case-sensitive string
- **user**—Searches for files belonging to the specified user
- **mtime**—The modify time; used for finding files x days old
- **atime**—Based on the number of days since last accessed
- **ctime**—Based on the number of days since the directory entry was last changed

A useful feature of the **find** command is its capability to execute another command or script on each and every entry normally returned to standard output.

For example, to find all MP3 files in the user's home directories and archive a copy into the root user's home directory, you could use this command:

```
find /home -iname *.mp3 -exec cp -f {} .\;
```

This command uses the **-exec** option, which accepts every line returned to standard output one by one and inserts the full path and filename between the curly brackets (**{}**). When each line of output is parsed and the command is executed, it reaches the **\;** at the end of the line and goes back to standard input for the next line. The last line of output is the last one with a command executed on it; it doesn't just keep going and error out.

Running multiple operators in a single command is possible, too. Just be sure not to get the values for one operator mixed up in the next. You could look for all MP3 files owned by a given user with the following command:

```
find /home -iname *.mp3 -user snuffy
/home/snuffy/bls - all for you.mp3
```

The **find** command is complex, and rather than bore you with more possible options, I've worked out a number of examples of how to use **find**:

**Key
Topic**

To find a file and execute **cat** on it, use

```
find /etc -iname fstab -exec cat {} \;
```

To delete all **core** files older than seven days, use the following:

```
find /home -mtime +7 -iname core -exec rm -f {} \;
```

To find all files on the system owned by **bob** and change the ownership to **root**, use

```
find / -user bob -exec chown root {} \;
```

To find all files by user **tjordan** and change his group, use this command:

```
find /data -user tjordan -exec chGRP users {} \;
```

For safety you can use **-ok** instead of **-exec** to be prompted for confirmation each time the command runs.

```
find /data -user tjordan -ok chgrp users {} \;
```

To find all inodes related to a hard link, use the command **find / -inum 123456**.

The **find** command's operators and the capability to execute commands on the search results will be covered on the exam. Practice all the examples you see here

and get inventive with the possibilities. Particularly watch out for the use of **-mtime** and its cousins: **-atime** and **-ctime**.

Which Command Will Run?

With the plethora of commands and executable scripts offered on a Linux machine, you need to know which of the possible commands will run when you type the name of it on the command line. This all depends on the contents of the **PATH** variable. This variable's contents are used as a sequentially read set of locations to search for executable objects.

The **which** command is used to determine the full path of commands that are queried from the **PATH** variable. To determine which command is indeed executed just by typing the name, run the following command:

```
which ls
alias ls='ls --color=tty'
      /bin/ls
```

As you can see, two entries were found that contain the **ls** command. The first is an alias, one that sets some color functions to the **ls** command; the other is the real command binary in **/bin/ls**.

When you execute a command, it finds the first available match, which might not be the one you wanted, as is the case with the **ls** command. To make it execute a physical binary and ignore any aliases that have been set, preface the command with a backslash (****), like so:

```
\ls
```

Try it again on a command that has two executables on the system, the **gawk** command:

```
which gawk
/bin/gawk
```

This returns a single entry, but there are multiple **gawk** commands on a Linux box. The first matching command found is returned by default, and only if you use the proper switch does it find all possibilities:

```
which -a gawk
/bin/gawk
/usr/bin/gawk
```

Researching a Command

When you need more information about a command than just which one will execute, try **whereis**. This command shows up to three possible bits of information,

including its binary files, the man page path, and any source files that exist for it. Here's its syntax:

```
$ whereis ls
ls: /bin/ls /usr/man/man1/ls.1.gz
```

Options for **whereis** include

- **-b**—Searches for binaries
- **-m**—Searches for manual entries
- **-s**—Searches for sources
- **-u**—Finds unusual or improperly documented entries

Key Topic

To find a file by name but not get all the entries that contain the name in the path, use the **whereis** command—not the **locate** command—because it finds the string in all elements of the path.

In Chapter 11, Customizing Shell Environments, you will learn how to extend the shell to make common tasks even easier. The **type** command will tell you if a command has been extended. To check what happens when you type **ps**:

```
$ type ps
ps is /bin/ps
```

The output of the **type** command above indicates that the **/bin/ps** application will be run if you type **ps**.

The **ls** command is often extended to show common options, such as to add color to the output:

```
$ type ls
ls is aliased to `ls --color=auto`
```

The output above shows that when you run **ls**, you actually get **ls --color=auto**. You can see all the possible variants of **ls** by using **type**'s **-a** option:

```
$ type -a ls
ls is aliased to `ls --color=auto`
ls is /bin/ls
```

The **-a** option shows that the shell knows about both an alias and a file on disk.

Linking Files

Links come in two varieties: symbolic and hard. (Symbolic links are often known as soft links.) Each has its own set of advantages and disadvantages. Sysadmins use links

for a multitude of purposes; chief among them is the need to make shortcuts on the system for users to access data without having to navigate multiple directory levels.

If you have users on your Linux systems, you need to have a single mount point accessible to multiple users. The options include having users navigate to the **/mnt/somemount** directory to save data or putting a link to that mount point in their home directories. You're much better off using a link for this task.

Symbolic Links

Symbolic links are used primarily to make a shortcut from one object to another. A symbolic link creates a tiny file with its own inode and a path to the linked file. Sym-links can span across filesystems and drives, primarily because a symlink has its own inode. Figure 5-1 shows the relationship between a symlink and the target file.

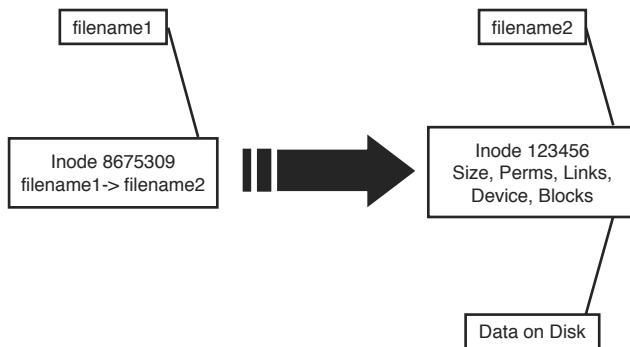


Figure 5-1 Symbolic link detail

For example, you might mount an external disk on the **/mnt/projdata** mount point and want each user to be able to access that remote share from her own home directory. You simply have to issue the following command in each user's home directory to accomplish this:

```
ln -s /mnt/projdata projdata
ls -l projdata
lrwxrwxrwx  1 root  root   13 Jan 26 12:09 projdata -> /mnt/
projdata
```

Notice that the listing for the new symlink shows exactly where the link points, and the permissions are set to the maximum so as to not interfere with the permissions on the target object.

Symbolic links always look like they have the same permissions, but don't try to change them. Changing permissions on a symlink changes the permissions on the target permissions instead.

Hard Links

A *hard link* is normally used to make a file appear in another place. A hard link is simply an additional name in a directory that points to the exact same inode and shares every aspect of the original file except the actual name (although the filename could be identical if in a different directory). Figure 5-2 shows the relationship between a hard link and the target file.

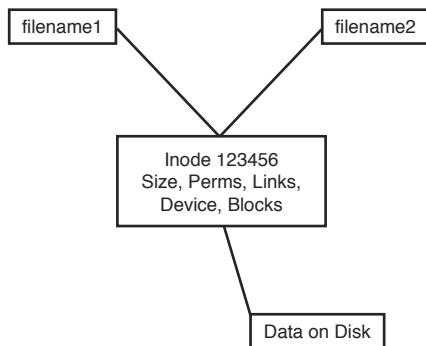


Figure 5-2 Hard link detail

For an example of using a hard link, consider the need to ensure that a frequently deleted file is easily restorable for a given user. The user, Jaime, travels a lot, but when he's in the office he seems to delete things a lot or claims the system has eaten his files. When Jaime is flying, you don't have any issues, so the problem must be the user's actions.

To anchor or back up an important file such as the company contact list in Jaime's home directory, you first must create a backup directory, something like **/backup**.

Then, you create a hard link from Jaime's **ccontactlist.txt** file to a file in the **/backup** directory, like so:

```

cd ~jaime
ln ccontactlist.txt /backup/home_jaime_ccontactlist.txt
ls -l ccontactlist.txt
-rw-r--r--  2 jaime  users    0 Jan 26 13:08 ccontactlist.txt
  
```

Notice that the file appears normal, but the number **2** for the link count lets you know that another name entry for this file exists somewhere.

Also notice that the listing for the new hard link doesn't show the target file or seem to refer to it in any way. Running the **stat** command on this file won't show you the other filename or seem to be aware of it outside the higher link count.

Key Topic

The name and location of a file are the only things about the file not stored in the inode. This appears on the exam in questions for this set of objectives.

Hard links can't be created if the target is on another filesystem, disk, or remote object. The need to associate multiple names to the same inode makes this impossible.

Be careful when changing the permissions and ownership on the hard-linked files because all name entries point to exactly the same inode. Thus, any changes are instantly made to what would appear to be multiple files but what, in reality, are only filenames.

To delete a file that has multiple hard links requires the removal of every hard link or the multiple names. To find all the links for a file, run the following command:

```
ls -i ccontactlist.txt
17392 ccontactlist.txt
find / -inum 17392
/home/jaime/ccontactlist.txt
/backup/home_jaime_ccontactlist.txt
```

Key Topic

NOTE On the exam, remember that a symlink is another actual file with its own inode. A large number of symlinks can therefore cause a problem for a filesystem, such as one that contains users' home directories. Too many inodes used can restrict you from using the storage space available. Run the **df -i** command to see what the statistics are.

Backup Commands

As an administrator you often are called upon to deal with file archives, which are one or more files that have been packaged into one file and optionally compressed.

There are several uses for archives:

- You want to send a few files to someone or copy them to another server and want to package and compress them.
- You need to back up a partition to other media in case a disk fails or the file is otherwise lost.

- You want to make a temporary copy of something before you make a change so you can restore it quickly if needed.
- You need to keep files around but in compressed format, such as for archiving old logs.

A number of backup options are available for Linux systems. Some are more useful than others, and some act on files, whereas others work best on partitions or disks as a unit.

Backup commands on the exams include the following:

- **cpio**
- **tar**
- **gzip** and **gunzip**
- **bzip2** and **bunzip2**
- **xz**

Using tar

The **tar** command is the workhorse of the archival world. The name comes from the term tape archive and goes back to the time when most backup was done to a local tape drive. You can think of **tar** as a pipeline that takes in a series of files and outputs a single file that is meant to be streamed to tape, but this output could be sent to a file on disk as well.

On the way through the pipeline you can do some transformations on the files such as chop up the output onto something that fits across multiple tapes, exclude files that weren't recently changed, or rewrite the directory names stored in the archive.

tar also provides the extraction options. You take a **.tar** file, also called a *tarball*, and run it through **tar** to get back a copy of the original files. It is possible to extract only certain files and manipulate the filenames.

The **tar** command also can use various compression commands, particularly the **gzip/gunzip** and **bzip2/bunzip2** commands by the use of special option characters. This has the effect of creating a compressed archive file, typically named **.tar.gz** for **gzip**-compressed files and **.tar.bz2** for **bzip2**-compressed files.

tar commands have an unusual syntax. The command is **tar**, followed by a dash (-), and then all the options concatenated together such as **xvjf**. After this is a list of zero or more filenames; the meanings depend on the options you chose.

Key Topic

The **tar** command has three main methods that act on files or **tar** archives; each has a corresponding letter that must be the first letter in the list of options:

- **c**—Creates an archive
- **t**—Tells you the contents of an archive
- **x**—Extracts files from an archive

The rest of the command can be optional, but some typical options are

- **v**—Be verbose by giving a list of files as they are processed.
- **j** or **z**—Compress or decompress with **bzip2** or **gzip**, respectively.
- **f**—The next word is the name of the file to operate on.

Figure 5-3 shows your choices graphically. We look at examples of each.



Figure 5-3 Picturing the tar options

When you're creating an archive with **tar**, you should think about what you want to archive, where you want the resulting archive to be created, and what compression if any you want to use.

Key Topic

To create a simple **tar** archive, the options you need are as follows:

```
tar -cf archive.tar /foo
```

In this example, the **-c** option signals **tar** to create the file specified after the **-f** option and specifies the directory you are archiving, which is the **/foo** directory. Note that you have to add the **.tar** suffix. By default the operation is recursive.

To create the same archive with **gzip** compression, you simply insert a **-z** option and use the letters **.gz** as the filename suffix:

```
tar -czf archive.tar.gz /foo
```

This creates a compressed archive file that uses the **gzip** compression algorithms. If you want slightly higher compression, use the **-j** option (instead of the **-z** option) for **bzip2** compression and create your archive with a suffix of **.bz** or **.bz2**.

Key Topic

You will likely see questions on the exam that test your knowledge of which compression command has the highest compression. For example, using **bzip2** generally results in a smaller archive file at the expense of more CPU cycles to compress and

uncompress. The **gzip** package is almost always part of the default installation of Linux while **bzip2** may not be.

To create a **tar** archive and see the filenames as they are processed use the **-v** option:

```
tar -cvf archive.tar /foo
```

This produces the following output:

```
tar: Removing leading `/' from member names
foo/
foo/install.log
foo/install.log.syslog
foo/.bash_logout
```

If given an absolute directory name to archive, **tar** strips the leading **/** from the full path of the objects in the archive. It would not be good if you could overwrite files in your **/usr** directory by extracting a file in an unrelated directory!

You may pass more than one directory or file to **tar**. For example, **tar -cf foo.tar bin var** creates an archive called **foo.tar** containing both the **bin** and **var** directories.

Taking Pity on the Unarchiver

It's considered proper and elegant to create **tar** archives by specifying a directory that contains the files to be archived, not just a bunch of files that are in the current directory. This means that when the files are untarred they show up in a single directory instead of in the current directory.

For example, create an archive of the **/etc** directory contents with the following command:

```
tar -cf etc.tar /etc
```

When you unarchive the **tar** file, by default it creates an **etc** directory in the current directory, which contains the entirety of the **/etc** directory you archived.

Contrast this with the nightmare that happens when you navigate to the **/etc** directory and create the archive from there with this command:

```
tar -cf /root/badetc.tar *
```

This archive file contains the same files as the previous one, except they aren't contained in a top-level **etc** directory—everything is in the top level of the archive.

Imagine what will happen to your system when you unarchive this file in the root user's home directory. You will have spewed approximately 2,400 files directly into the root user's home directory!

It really does matter where you are in the filesystem and which path options you use when you create or expand an archive file. It's best practice to use absolute path-names.

To solve the problem of 2,400 files polluting your root user's home directory, use the following command, where **badetc.tar** is the offending archive file:

```
tar -tf badetc.tar | xargs rm -rf
```

This command produces a list of the paths and filenames of files in the archive and uses the **xargs** command to feed each line of output as a filename specification to the **rm -rf** command, removing all the files and directories expanded from the **badetc.tar** file.

Useful Creation Options

A number of other options can be used for creating **tar** archives. Here is a list of the more useful and testable ones:

- **-b**—Sets the block size to fit the media to which you are archiving. This is necessary for some tape devices.
- **-M**—This specifies multiple archive targets or spreads a large archive across multiple tapes or media.
- **-g**—Creates a new format incremental backup (only those that have changed since the last full or incremental).
- **-I**—Stays on the local filesystem; it's used to keep from backing up the entire NFS network by accident.
- **-L**—This is followed by a number that reflects 1024 bytes, so **-L 500** equals 500KB. (It's used for setting the tape length so multiple tapes can be used for an archive.)
- **--remove-files**—This is dangerous because the specified files are removed from the filesystem after they have been added to the archive!

Listing Archive Files

An underrated option, listing is something that typically is used after you don't get the results you want or realize what you've just done and want to confirm how hard it is going to be to clean up.

To tell you the contents of a **tar** archive, use the following command:

```
tar -tf archive.tar
```

This produces the output shown here:

```
etc/
etc/sysconfig/
etc/sysconfig/network-scripts/
etc/sysconfig/network-scripts/ifup-aliases
etc/sysconfig/network-scripts/ifcfg-lo
```

To list an archive that uses compression, simply insert the necessary letter between the **-t** and the **-f** options, such as the **bzip2 -j** option shown here:

```
tar -tjf archive.tar.bz2
```

This produces the following output:

```
etc/
etc/sysconfig/
etc/sysconfig/network-scripts/
etc/sysconfig/network-scripts/ifup-aliases
etc/sysconfig/network-scripts/ifcfg-lo
```

To list an archive and see the file details for its contents, you add the **-v** option to the existing command to see an output of the details:

```
tar -tvjf archive.tar.bz2
```

This returns output similar to the following:

```
drwxr-xr-x root/root          0 2015-02-10 03:46 etc/
drwxr-xr-x root/root          0 2015-01-31 10:09 etc/sysconfig/
drwxr-xr-x root/root          0 2014-11-10 22:13 etc/sysconfig/network-
scripts/
```

Key Topic

When you create an archive with the **-v** option, a list of the files being archived is shown onscreen. When you unarchive an archive with the **-v** option, it shows a similar list of the files being unarchived.

It's only when you list an archive with the **-v** option that you get the type of output that approximates an **ls -l** command being run on the archive contents. This is an exam topic, so be ready for it.

Using cpio

The **cpio** command appears extensively in the Level 2 LPI objectives. This level of the exam might ask you about the **cpio** command at only the simplest levels, such as knowing that it exists, how it works in general terms, and whether it can be used to back up a Linux system.

The **cpio** command actions all treat the filesystem as the home base. If you are copying out, it's from the filesystem out to another file. The same is true with copying in—it's from a file into the filesystem.

The **cpio** command has three options for acting on files and filesystems:

- **-o** or **--create**—This copies files to an archive using a list of files typically created by the **find** command.
- **-i** or **--extract**—This copies files into the filesystem from an archive or a list of the archive contents.
- **-p** or **--pass-through**—This copies files from one directory tree to another without the use of an archive, essentially performing the same function as the **cp -r** command.

Key Topic

The **cpio** command accepts a list of files in a one-file-per-line format and uses this list to send the archived files to either the standard output or an archive file you specify.

cpio supports a variety of archive formats, including binary, ASCII, crc, and tar, to name the most relevant.

An example of creating a **cpio** archive from the files in the current directory is shown here:

```
find . "*" | cpio -o > archive.cpio
```

This outputs the list of files found by this particular **find** command, with the **cpio** command taking the entirety of the files and sending them to the **archive.cpio** file by redirecting standard output to the file.

The **cpio** command doesn't accept a list of files to archive on the command line like the other utilities you've seen so far. Instead, it reads the names of the files from the standard input or console. So be aware that using either the **find** or **ls** command is necessary to feed **cpio** a list of filenames.

For example, if you needed to archive all the files that have an extension of **.txt** in the current directory to a **cpio** archive named **txt.cpio**, you would use the following command:

```
ls *.txt | cpio -o > txt.cpio
```

Notice that you're redirecting the output of **cpio** to a file rather than letting it write the file itself. Therefore the filename is up to you, and if you want a **cpio** file extension, you need to add it yourself.

Compression Utilities

Whereas the **tar** command is used to gather files and put them in a container, the **gzip**, and **bzip2** commands are used to compress that container. Used by themselves, they act on each file they find and replace that file with a compressed version that has an extension that indicates the file is compressed.

The **gzip** and **bzip2** compression utilities compress files and are similar in their functions and operations. The main difference is that **bzip2** offers slightly better compression than **gzip**, but **gzip** is much more widely used.

These commands replace the original file with a new file that has an additional extension, so don't delete the `.gz` or `.bz2` files that you create. They are the original files in a compressed wrapper!

To compress all the files in the current directory with **gzip** or **bzip2**, use this command:

```
gzip *
```

This replaces all the regular files (not the directories or their contents) in the current directory with the original filenames plus a `.gz` extension. So, if you had two files named `file1` and `file2` in the directory, they would be replaced with

```
file1.gz
file2.gz
```

To uncompress these files, just do the exact opposite of the compression:

```
gunzip *
```

This restores the original files.

Using **bzip2** produces the same sort of results. You can issue the following command in the same directory:

```
bzip2 *
```

You would then have the following two files:

```
file1.bz2
file2.bz2
```

To uncompress these files, issue this command:

```
bunzip2 *
```

This restores the files to their original states.

xz is a third option for compressing files just like **bzip2** and **gzip**. It is newer, and in some cases has better performance than **bzip2** at a cost of more memory. Files are compressed with one of **xz**, **xz -z**, or **xz --compress**, and decompressed with one of **unxz**, **xz -d**, **xz --uncompress**, or **xz --decompress**.

The **.xz** file extension indicates that a file was compressed with **xz**. To uncompress **foo.xz** you would run **xz -d foo.xz**, and would be left with an uncompressed file called **foo**.

Watch for questions that ask about why you would use either **gzip** or **bzip2** for a particular compression task. **bzip2** offers slightly better compression at the expense of increased CPU cycles. **gzip** is faster but doesn't compress as well. **gzip** also has a recursive option (**-r**) that compresses all files in a directory.

Summary

In this chapter you learned about the Linux File System Hierarchy Standard (FHS) and what it means for laying out partitions. You also learned how to find files in real time with the **find** command, and through a database lookup with the **locate** command. This chapter also covered the **cp**, **mv**, and **touch** commands for copying, moving, and updating files, along with the proper use of file globs for matching files on the command line.

Finally you learned about the various archival and compression utilities that Linux makes available to you.

Exam Preparation Tasks

As mentioned in the section “How to Use This Book” in the Introduction, you have a couple of choices for exam preparation: the exercises here, Chapter 21, “Final Preparation,” and the practice exams on the DVD.

Review All Key Topics

Review the most important topics in this chapter, noted with the Key Topics icon in the outer margin of the page. Table 5-3 lists a reference of these key topics and the page numbers on which each is found.

Key
Topic**Table 5-3** Key Topics for Chapter 5

Key Topic Element	Description	Page Number
Paragraph	FHS documentation about what goes on the root volume	113
Paragraph	The use of the <code>/usr</code> and <code>/usr/local/</code> directories	114
Paragraph	Relative pathnames and <code>.</code> (period character)	116
Paragraph	Long listing format (<code>-l</code>) to see permissions	117
Paragraph	Using the touch command	120
Paragraph	Using a glob to avoid copying into a directory incorrectly	122
Paragraph	Locate needs the database refreshed periodically	129
Paragraph	Locate searches whole names	130
Paragraph	Examples of find usage	131
Paragraph	When to use whereis versus locate	133
Paragraph	The filename is not stored in the inode	136
Note	Symlinks consume inodes	136
Paragraph	The order and function of tar 's options	138
Paragraph	Creating a tar archive	138
Paragraph	bzip2 has the highest compression rate	138
Paragraph	The <code>-v</code> option to tar	141
Paragraph	cpio accepts its files from the standard input	142

Define Key Terms

Define the following key terms from this chapter and check your answers in the glossary:

File System Hierarchy Standard, relative path, absolute path, hard link

Review Questions

The answers to these review questions are in Appendix A.

1. You are installing a customized server and need to strip the root filesystem down to the essentials only. According to the FHS 2.3, which of the following are considered optional on the root (/) filesystem? (Choose two.)
 - a. /root
 - b. /usr
 - c. /tmp
 - d. /home

2. One of your programmers has produced an order entry system that will be shared among your users from a central file server. What is the appropriate directory to place this program and its associated files in?
 - a. /usr/local/bin
 - b. /usr/local
 - c. /usr/share
 - d. /opt

3. Which of the following is a true statement about files on a default Linux system? (Choose all that apply.)
 - a. Filenames can start with a number.
 - b. Filenames can contain multiple periods.
 - c. Filenames can contain spaces.
 - d. Filenames can contain ampersands.
 - e. Filenames can contain backslashes.

4. You find a string in a shell script that contains the following command:

```
cp /data/*.doc ~tarfoo
```

What is the meaning of the characters ~tarfoo?

- a. A special function named tarfoo
- b. A directory named tarfoo in your home directory
- c. The tarfoo user's home directory
- d. The /data/tarfoo directory

5. You are currently in the directory `/home1/user1/subdir1` and need to navigate to the directory `/home12/user3`. Which of the following commands will accomplish this?
 - a. `cd home12/user3`
 - b. `cd ~/user3`
 - c. `cd ../../home12/user3`
 - d. `cd ../.././home12/user3`

6. You have a directory named `/dir1` that contains subdirectories and regular files. You want to replicate this directory structure exactly into an existing directory named `/dir2`. Which of the following commands accomplish this? (Choose all that apply.)
 - a. `cp --contents dir1/ /dir2`
 - b. `cp -r /dir1/* /dir2`
 - c. `xcopy /dir1 /dir2`
 - d. `cp -r /dir1 /dir2`

7. You are currently in the `/bbb` directory and want to move the contents from the `/ccc` directory to this one. What is the shortest command that will accomplish this?
 - a. `mv /ccc/*.* .`
 - b. `mv ../ccc/*.* .`
 - c. `mv /ccc/* .`
 - d. `mv /ccc/ /bbb`

8. Which option to the `mkdir` and `rmdir` commands allows you to create a nested subdirectory tree?

Example:

```
/dir1/dir2/dir3/dir4
```

- a. `-c`
- b. `-n`
- c. `-d`
- d. `-p`

9. You are the sysadmin of a busy server and need to save space on your /home partition. You need to remove all files named **core** that are older than seven days in the users' home directories, without receiving any prompts.
- a. `find /home -mtime +7 -name core -exec rm -f {} \;`
 - b. `find ~ -mtime +7 -name core -exec rm -f {} \;`
 - c. `find /home -mtime -7 -name core -exec rm -f {} \;`
 - d. `find /home -older 7d -name core -exec rm -f {} \;`
10. Which of the following situations would prevent you from creating a hard link?
- a. The link spans filesystems.
 - b. The source of the link is a hidden file.
 - c. The source of the link is a device file.
 - d. The source of the link is a directory.
 - e. The destination contains special characters.
11. How would you back up Rebecca's home directory using the best compression available?
- a. `cd /home; tar -czf rebecca.tgz rebecca`
 - b. `find ~rebecca | tar -cjf - > rebecca.tar.bz2`
 - c. `tar -cjf rebecca.tar.bz2 ~rebecca`
 - d. `tar -xjf rebecca.tar.bz2 ~rebecca`

This page intentionally left blank



Index

Numbers

- 1 boot option, 32
- 2 boot option, 32
- 3 boot option, 32
- 4 boot option, 32
- 5 boot option, 32
- 100 entry, */etc/passwd* file, 424
- 500 entry, */etc/passwd* file, 424

A

- access times setting (*xinetd* command), 580
- accounts, 422
 - group, 425-427
 - /etc/group* file entries, 427
 - adding, 431
 - GIDs, 422-423, 426
 - modifying, 432-433
 - passwords, 427-428
 - primary, 425
 - removing, 434-435
 - secondary, 426
 - UPG (*User Private Group*), 426-427
 - limiting, 440
 - passwords, changing, 437-440
 - root, access, 574-576
 - security, Shadow Suite, 435-437

user

- adding, 428-430
- entries in */etc/passwd* file, 423-424
- modifying, 431-432
- removing, 433-434
- special login files, 424-425
- UIDs (*User IDs*), 422-423, 424

actual commands, 97

adding

- group accounts, 431
- user accounts, 428-430

ADDITIONAL section (*dig* query), 558

address class ranges, networks, 534-535

address classes, networking, 535

address schema, IP (*Internet Protocol*), 544-545

ad-hoc jobs, running, 456

- batch command, 458-459
- at command, 456-458

aging passwords, 438-439

aliases, 97

- creating, MTAs (*Mail Transfer Agents*), 489

- MTAs (*Mail Transfer Agents*), 490
- shells, 93, 323

always on services, 576

anacron system, 447, 455-456

ANSWER section (*dig* query), 558

apm boot option, 32
apt-cache command, 66
apt-get command, 67
aquota.group file, 277
aquota.user file, 277
archives, tar, creating, 137-141
arguments, accepting, 357-358
ash shell, 314
at command, 447, 456-458
atq command, 457
atrm command, 457
AUTHORITY section (dig query), 558
autoconfigure, IPv6, 545
automatically mounting filesystems, 273
automount units (systemd), 36
awk command, 168

B

backup commands, 136-137
bash shell, 93

- history feature, 103-104
- setting options, 104-105

batch command, 458-459
bin directory (FHS), 112
bin subdirectory (/usr directory), 114
/bin/bash entry, /etc/passwd file, 424
/bin/false file, 424
binaries, software, 54
binary files, viewing, 173-174
binary shell, 93
blkid command, 270
blocks, processes, 199
boot directory (FHS), 112
boot events, logging, 45-46
boot loaders, 17-21, 30-31

GRUB (Grand Unified Boot Loader),
 17-18, 31
GRUB2, 18

- command line*, 19-20
- configuring*, 20-21
- installing*, 18-19

LILO, 31
Loadlin, 31
SYSLINUX, 31

boot process, 30

command prompts, 32
 common commands, 32-33
 phases, 30
 systemd, 34-36, 38-42

- changing runlevels*, 41-42
- determining default runlevels*, 40-41
- managing system runlevels*, 40
- runlevels*, 37
- setting default runlevels*, 41
- targets*, 37-38
- units*, 36
- Upstart system initialization scheme*,
 39-40
- wants and requires*, 38

daemons, 35

SysVinit, 33-34

Bounce Keys, X Window System, 411

Bourne Again shell, 314

Braille Display, X Window System, 413

breaking long command line, 98-99

broadcast addresses, 538

btrfs filesystem, 256

buffers

processes, 199

vim text editor, 229-230

built-in commands, 97

buses, peripheral/CPU
communication, 7
bzip2 command, 143-144

C

cached, 199
remote package repositories, 65-66
case statements, 351
cat command, 164
cd command, 115-116
certifications, 610-611
chage command, 438-439, 596
character encoding, 329-330
checking filesystems, 263-266
chgrp command, 306-307
child environment, setting variables
from, 316-317
chmod command, 295-297, 303
chown command, 305-306
classes.conf file, 482
classifying data, FHS (Filesystem
Hierarchy Standard), 113-114
clauses, WHERE, SQL (Structured
Query Language), 374-376
cleaning up SQL queries, 381
client utilities, SSH (Secure Shell),
592-596
clocks, 500
hardware, 500
synchronizing, 504-505
system, 500
columns, cutting, 168
command line, 90-91, 97
commands
breaking long, 98-99
completion, 99

controlling execution, 100-101
executing multiple, 161-163
structuring, 98
types, 97

CUPS (Common Unix Printing
System), 477

environment variables, 101-104

GRUB2, 19-20

input/output streams, 154-158

pipes, 159-161

printing from, 478

rebooting from, SysVinit, 43

shutting down from, SysVinit, 42

Command mode, vim text editor, 223

command prompts, boot process, 32

commands

actual, 97

aliases, 97

apt-cache, 66

apt-get, 67

at, 456-458

atq, 457

atrm, 457

awk, 168

backup, 136-137

batch, 458-459

blkid, 270

built-in, 97

bzip2, 143-144

cat, 164

cd, 115-116

chage, 438-439, 596

chgrp, 306-307

chmod, 295-297, 303

chown, 305-306

controlling execution, 100-101

cp, 121-123, 125
cpio, 141-142
crontab, 447-448
cupsaccept, 480-481
cupsdisable, 481
cupsenable, 481
cupsreject, 480-481
cut, 168
date, 328, 500-503
dd, 126-127
debugfs, 257, 268-269
df, 275-276
dhclient, 554, 556-557
dhcpcd, 554, 556-557
dig, 554, 557-559
dmesg, 45-46
dpkg, 59-60
du, 274-275
dumpe2fs, 262-263
echo, 119
edquota, 277
egrep, 184-185
env, 319-320
executing multiple, 161-163
expand, 167-168
export, 316
fdisk, 246-250
features, 269
fgrep, 184-185
file, 118-120
filters, 165-174
find, 130-132, 586-587, 596
fmt, 174-175
formatting, 174-175
 grep command, 178-184
 sed (stream editor) command, 175-176
 translating files, 175-176
free, 198-201
fsck, 263-266, 269
fstab, 270
functions, 97
fuser, 596
gdisk, 250-254
getent, 557-559
gpasswd, 428
gpg -gen-key, 587-589
grep, 178-187
groupadd, 431
groupmod, 432-433
grpquota, 277
grub-mkconfig, 21
gzip, 143-144
halt, 42
head, 170-171
history, 103-104
host, 554, 557-559
hostname, 350, 554, 559
hwclock, 503-504
ifconfig, 554-555
ifdown, 554
ifup, 554
inetd, 576-578, 581
install, 64-65
ip, 554
ip addr show, 547
ipconfig, 545-546
join, 168-169
journalctl, 515-516
kill, 44-45
killall, 44-45, 204
last, 596
less, 159
locale, 330-334

locate, 128-130
 logdump, 269
 logger, 512
 lp, 478
 lpq, 479
 lpr, 478
 lprm, 479-480
 lpstat, 478-479
 ls, 116-118, 132, 269
 lsblk, 9
 lscpu, 9
 lsdev, 9
 lsmod, 8
 lsof, 596
 lspci, 9
 lsraid, 9
 lsscsi, 9
 lsusb, 9
 mkdir, 127-128
 mkfs, 260-262
 mount, 270, 273
 mv, 124-126
 n, 252-253
 netstat, 554, 559-562
 newgrp, 428
 nice, 207
 nl, 166-167
 nslookup, 554
 ntpd, 507-508
 od, 173-174
 open, 269
 parted, 254-255
 paste, 168-169
 passwd, 439, 596
 ping, 554, 562-563
 pkill, 204-205
 poweroff, 42
 pr, 174-175
 ps, 196-198
 pstree, 197-198
 pump, 554, 556-557
 pwd, 269
 quit, 269
 quota, 277, 280-282
 quotacheck, 277
 quotaon, 277
 quotoff, 277
 renice, 207-208
 route, 554-555
 rpm, 70-71
 scp, 591, 593
 screen, 210-213
 scripts, 97
 sed, 176-178
 seq, 354-355
 set, 317-318
 shell script, 344
 combining multiple tests, 351
 performing math, 345-346
 testing files, 348-349
 testing integers, 349-350
 testing strings, 349-350
 use output of another command,
 344-345
 shutdown, 42
 sort, 165-166
 split, 172-173
 SQL (Structured Query Language),
 372-373
 ssh, 591-592
 ssh-add, 591
 ssh-agent, 591
 sshd, 591

- stat, 124
- stats, 269
- structuring, 98
- su, 574-575
- substitution, 162-163
- sudo, 575-576
- swap, 245
- tac, 173
- tail, 171-172
- tar, 137-141
- tcpdump, 554, 565
- tee, 163
- test, 348
 - combining multiple tests, 351*
 - syntax, 348-349*
 - testing integers, 350*
 - testing strings, 349-350*
- top, 208-209
- touch, 120-121
- tr, 175-176
- traceroute, 554, 563-564
- tune2fs, 266-267
- ulimit, 440, 596
- undelete, 269
- uniq, 169-170
- unset, 317-318
- uptime, 201-202
- useradd, 428-430
- userdel, 433-434
- usermod, 431-432, 596
- usrquota, 277
- vi text editor, running external, 234
- vim text editor, cut/copy/paste, 228-229
- w, 596
- whereis, 132-133
- which, 132

- who, 596
- xargs, 163-164
- xfst, 267-268
- xinetd, 576-583
- xwininfo, 407-408

comments, SQL (Structured Query Language), 372

Common Unix Printing System (CUPS). *See* CUPS (Common Unix Printing System)

communication, peripherals, 7

compatibility, peripherals, 7

completion shell, 93

component groups, systemd, 35

compression utilities, 143-144

conditions

- scripts, 346-347

- SQL (Structured Query Language), multiple, 376-377

conffiles section (.deb file), 59

configuration

- cron system, crontabs, 447-448

- CUPS (Common Unix Printing System), 470-474

- default gateways, 550-551

- disk quotas, 278-280

- GRUB2, 20-21

- interfaces

- Debian, 548-549*

- Red Hat, 547-548*

- network utilities, 553-554

- dbclient command, 556-557*

- dhcpcd command, 556-557*

- dig command, 557-559*

- getent command, 557-559*

- host command, 557-559*

- hostname command, 559*

- ifconfig* command, 555
- netstat* command, 559-562
- ping* command, 562-563
- pump* command, 556-557
- route* command, 555
- tcpdump* command, 565
- traceroute* command, 563-564
- ntpd daemon, 506-507
- systemd, 512-514
- TCP wrappers, 583-585
- YUM packages, 84-86
- configuration files**
 - inetd, 577-578
 - RPM packages, querying, 79
 - xinetd, 578-581
 - Xorg, 402-405
- console, 210-211**
 - locking, 213
 - multiple sessions, 210-211
- consoled daemon (systemd), 35**
- control section (.deb file), 59**
- convenience crontabs, 454**
- copy command, vim text editor, 228-229**
- copying files and directories, 121-123**
- cp command, 121-123, 125**
- cpio command, 141-142**
- CPUs, peripherals, addressing, 7-8**
- cron system, 447**
 - configuring crontabs, 447-448
 - convenience crontabs, 454
 - crontab command, 447-448
 - files, 452-453
 - finishing crontab, 450
 - making multiple matches, 449
 - matching time, 448-449
 - nicknames extension, 452
 - output, 451-452
 - PATH, 450-451
 - restricting access, 454-455
 - spelling out month and day names, 449
 - step values, 450
 - system crontabs, 453
- crontab command, 447-448**
- C-shell, 314**
- CUPS (Common Unix Printing System), 468**
 - Administration menu, 476
 - command-line tools, 477
 - configuration files, 482
 - configuring, 470-474
 - daemon, 468
 - daemons, configuring, 481
 - Jobs List, 477
 - legacy printing tools, 477-480
 - maintenance, 474-475
 - maintenance and administration
 - pull-downs, 476
 - pipeline, 468-470
 - print jobs, rejecting, 480-481
 - printer state, 475
 - printers, enabling/disabling, 481
 - troubleshooting printing, 482-483
- cupsaccept command, 480-481**
- cupsd.conf file, 482**
- cupsd.conf.default file, 482**
- cupsdisable command, 481**
- cupsenable command, 481**
- cupsreject command, 480-481**
- custom network masks, 538-541**
- cut command, 168**
 - vim text editor, 228-229
- cutting columns, 168**

D

daemons

- CUPS (Common Unix Printing System), 468
- ntpd, configuring, 506-507
- systemd, 35

data formats, transforming, 126-127

databases, 368

- key-value, 368-369
- relational, 369-370
- RPM, 68-69
- schemaless, 370-371
- SQL (Structured Query Language), 371
 - advanced joins*, 381-384
 - cleaning up queries*, 381
 - commands*, 372-373
 - comments*, 372
 - creating tables*, 388-389
 - deleting data*, 388
 - grouping data*, 386-387
 - inserting data*, 387
 - keywords*, 372-373
 - left versus right joins*, 384
 - limiting results*, 378
 - multiple conditions*, 376-377
 - multiple tables*, 378-379
 - NULL, 384-385
 - selecting data*, 373-374
 - semicolons*, 372
 - sorting results*, 377-378
 - SQLite, 371-372
 - subselects*, 385-386
 - updating data*, 388
 - WHERE clause*, 374-376
 - writing queries with joins*, 379-380

date command, 328, 500-503

date specifier (at command), 457

dd command, 126-127

.deb files. *See* local packages

Debian, 50

- interface configuration, 548-549
- package management, 58-59
 - caches*, 65-66
 - dependency issues*, 60-61
 - graphical managers*, 67
 - installing packages*, 59-60
 - installing remote packages*, 64-65
 - querying packages*, 61-63
 - reconfiguring packages*, 61-63
 - removing packages*, 60
 - removing remote packages*, 67
 - system upgrade*, 66-67

debugfs command, 257, 268-269

debugging filesystems, 268-269

default gateways

- configuring, 550-551
- viewing, 550

default network mask, 536-537

default permissions, files, 303-305

defaults, useradd command, 429

Delete Printer command

(Administration menu), 476

deleting

- data, SQL (Structured Query Language), 388
- Debian packages, dpkg command, 60
- group accounts, 434-435
- objects, 128
- remote Debian packages, apt-get command, 67
- RPM packages, 75-76
- text, 227-228
- user accounts, 433-434

- dependencies, RPM packages, querying, 80**
- dependency issues, Debian packages, 60-61**
- descriptors, files, 154**
 - stderr, 155-156
 - stdin, 154
 - stdout, 154-155
- design, scripts, 343**
- desktops, X Window System, 402**
- detaching processes, 212-213**
- dev directory (FHS), 112**
- device units (systemd), 36**
- devices, 6**
 - compatibility, 7
 - enumerating, 7-8
 - integrated, 10-11
 - procs, 8-10
- df command, 275-276**
- dhclient command, 554, 556-557**
- dhcpcd command, 554, 556-557**
- dig command, 554, 557-559**
- direct memory access (DMA), CPU/peripheral addressing, 8**
- directories**
 - copying, 121-123
 - creating and removing, 127-128
 - FHS (Filesystem Hierarchy Standard), 112
 - /usr directory, 114*
 - inodes, 258-259
 - permissions, setting SGID bit, 301-302
- disable setting (xinetd command), 580**
- disabled services, 576**
- disk partitioning tools, 245**
 - fdisk, 246-250
 - gdisk, 250-254
 - parted, 254-255
- disks**
 - /dev/sda, 21
 - disk quotas, 277-282
 - filesystem mounts, 16
 - laying out, 11-17
 - PIBS (Performance, Integrity, Backup, Security), 14*
 - LVM (Logical Volume Manager), 14-16
 - partitions, 11-12
 - swap, 16-17*
 - tools, 245-255*
 - quotas, 277-282
 - commands, 277*
 - configuring, 278-280*
 - grace period, 278, 281*
 - hard limit, 277, 280-281*
 - obtaining information, 281-282*
 - soft limit, 277, 280-281*
 - root filesystem, 12-14
 - space, inodes, 259-260
- display managers, X Window System, 408-409**
- distributions, 610-611**
- DMA (direct memory access), CPU/peripheral addressing, 8**
- dmesg command, 45-46**
- DNS (domain name system), MTAs (Mail Transfer Agents), 487**
- documents**
 - replacing, 231
 - searching, 231
- dpkg command**
 - installing packages, 59-60
 - removing packages, 60
 - viewing packages, 58-59
- du command, 274-275**

dumpe2fs command, 262-263

dynamic content, shells

adding, 325-326

creating new users, 326-327

E

echo command, 119

editing shell, 93

EDITOR environment variables, 448

edquota command, 277

egrep command, 184-185

emacs option (bash), 105

email

flow, 484

language, 485-486

MTAs (Mail Transfer Agents), 484-485

aliases, 490

committing changes, 489-490

creating aliases, 489

DNS (domain name system), 487

forwarding, 489

Linux, 486-487

mail server, 488-489

managing queues, 491-492

MDAs (Mail Delivery Agents), 488

user-defined forwarding, 491

encodings, converting files between, 334

encryption

GnuPG keys, 587-590

passwords, Shadow Suite, 435-437

enumerating, peripherals, 7-8

env command, 319-320

environment variables, 315

EDITOR, 448

error checking, filesystems, 263-266

error code, returning, 357

etc directory (FHS), 112

/etc/group file, 427

/etc/init.d directory, 596

/etc/inittab file, 596

/etc/login defs file, 425

/etc/motd file, 425

/etc/nonlogin file, 424

/etc/passwd file, 596

entries, 423-424

Shadow Suite, 435-437

/etc/security file, 425

/etc/shadow file, 596

/etc/usertty file, 425

events, boot, logging, 45-46

examples

/etc/adjtime (18-3), 504

Actual Hierarchy of the Boot Process (2-2), 39

Contents of an /etc/xinetd.conf File (20-1), 579

Contents of an /etc/xinetd.d Service File (20-2), 580

Create a Partition with the Label “MyData” Going from the Beginning of the Drive to 7GB (9-8), 254-255

Create Partition for Primary Partition Number 2 (9-2), 248

debugfs Command (9-9), 257

Default Sources List File (3-3), 65

Default yum.conf Configuration for Fedora (3-10), 84-85

Demonstration of Changing GID (15-2), 433

Demonstration of /etc/skel (15-1), 430

Demonstration of SUID (10-1), 300

dumpe2fs (9-11), 262-263

- Example journald.conf (18-11), 519
- Example ntp.conf (18-4), 506
- Example of a Variable (11-1), 315
- Example of an ifconfig Command's Output (19-2), 546
- Example of Contents of the /etc/network/interfaces File (19-4), 549
- Example of dig Command Output (19-5), 558-559
- Example of ip addr show Command Output (19-3), 547
- Example of the /etc/services File (19-1), 542-543
- Example of traceroute Command Output (19.7), 563-564
- Example of xwininfo Output (14-22), 407
- Example Xorg Configuration File (14-1), 403-404
- Examples Using a Glob (5-1), 119-120
- f Forces a Check (9-13), 266
- Finding a Package's Configuration Files (3-5), 79
- Finding Out What's Inside a .deb File (3-1), 58-59
- Individual Repository's Configuration from Fedora (3-11), 85
- Installing a Package from a Remote Repository (3-7), 81-82
- Installing a Remote Package (3-2), 64
- Installing GRUB2 to an Alternate Location (1-2), 18-19
- List the Partitions (9-5), 252
- Listing Packages That Match a Wildcard (3-8), 83-84
- logrotate Configuration (18-12), 521
- Looking at the Logs Verbosely (18-10), 517-519
- lsdev Command (1-1), 9
- Mail Queue (17-8), 491
- Making the Filesystem with xfs (9-14), 267-268
- mkfs and Options (9-10), 261-262
- Moving Files to a New Name (5-3), 124
- Output of the gpg --gen-key Command (20-3), 588
- Output of the gpg --gen-key Command (20-4), 588-589
- Output of the gpg --gen-key Command (20-5), 589
- Partition Verification (9-7), 253-254
- Printing from the Command Line (17-1), 478
- Querying a Package for Information (3-4), 77-78
- Querying All the Service's Logs (18-9), 517
- Querying an NTP Server's Peers (18-5), 507
- Querying the Dependencies of a Package File (3-6), 80
- Querying the Journal for sshd Logs (18-8), 516-517
- Querying the Printer for Status (17-3), 479
- Rejecting Jobs (17-5), 480-481
- Removing a Print Job (17-4), 480
- Running Commands and Functions in Your Prompt (11-6), 325
- Running the stat Command on file1 (5-2), 124
- Sample Check on a Filesystem (9-12), 265
- Sample /etc/syslog.conf (18-6), 512-513
- Script to Demonstrate BASH_SUBSHELL Levels (11-5), 319
- Searching YUM Repositories for Packages That Match a Concept (3-9), 84

- Setting the Time in Various Formats (18-1), 501
 - Shell Script to Demonstrate Subshells (11-4), 319
 - Showing the Status of All Printers (17-2), 478
 - SMTP Conversation Between Two MTAs (17-6), 485
 - String Matched as if It Were a File Glob (12-2), 352
 - Theoretical Hierarchy of the Boot Process (2-1), 38
 - Updating aquota.* with quotacheck -avugc (9-15), 279
 - Using case Instead of if/elif/else (12-1), 351
 - Using date in a Shell Script (18-2), 502
 - Using /etc/skel (11-7), 326-327
 - Using gdisk -l to View Partition Information (9-3), 251
 - Using gdisk to Convert an Existing MBR Partition Table to GPT (9-4), 252
 - Using locale (11-10), 333
 - Using locale-gen (11-9), 332
 - Using n to Create a Partition (9-6), 253
 - Using noclobber (11-3), 318
 - Using the locale -a Command to See the Locales Installed on Your System (11-8), 331
 - Using the shift Keyword (12-3), 357-358
 - Using unset to Destroy a Variable (11.2), 317
 - Verify the Partition Was Created (9-1), 247
 - Viewing the Unfiltered Log with journalctl (18.7), 515-516
- exams (LPI)**
- certifications and distributions, 610-611
 - objectives, 604, 606-608
 - preparation, 603-605, 610, 616
 - software installation*, 611
 - studying*, 608
 - studying don'ts*, 609-610
 - VMs (virtual machines)*, 609
 - question amount and time, 605
 - question types, 611-612
 - choose all that apply*, 613-614
 - choose two/choose three*, 613
 - fill in the blank*, 615-616
 - single answer multiple choice*, 612
- executing multiple commands, 161-162**
- execution, commands, controlling, 100-101**
- expand command, 167-168**
- export command, 316**
- expressions, grep command, 185-187**
- ext2 filesystem, 255**
- ext3 filesystem, 255**
- ext4 filesystem, 255**
- extending shells, 320**
- adding dynamic content, 325-326
 - aliases, 323
 - creating new users, 326-327
 - functions, 323-324
 - global and user settings, 320
 - login session, 321
 - non-login session, 321
 - PATH variable, 322-323
 - PS1 variable, 324-325
 - PS2 variable, 326
- extensions, files, 115**
- external commands, vi text editor, running, 234**

F

fallback locales, 331

fdisk command, 246-250

features command, 269

fgrep command, 184-185

FHS (Filesystem Hierarchy Standard), 54, 112-115

classifying data, 113-114

commands

- backup*, 136-137
- cpio*, 141-142
- researching*, 132-133

common directories, 112

data formats, transforming, 126-127

directories

- /usr*, 114
- copying*, 121-123
- creating and removing*, 127-128
- listing*, 116-118

files

- compression utilities*, 143-144
- copying*, 121-123
- determining type*, 118-120
- finding*, 130-132
- linking*, 133-136
- listing*, 116-118
- locating*, 128-130
- touching*, 120-121

navigation commands, navigation, 115-116

objects

- moving*, 123-126
- removing*, 128

which command, 132

file command, 118-120

file permissions, Shadow Suite, 437

files

aquota.group, 277

aquota.user, 277

binary, viewing, 173-174

compression utilities, 143-144

converting encodings, 334

copying, 121-123

.deb, managing, 58-64

descriptors, 154

- stderr*, 155-156
- stdin*, 154
- stdout*, 154-155

determining types, 118-120

/etc/group file, 427

/etc/passwd, entries, 423-424

extensions, 115

finding, 130-132

group ownership, changing, 306-307

hidden, 115

inodes, 256-258

linking, 133-134

- hard*, 134-135
- symbolic*, 134-135

listing, 116-118

- archive*, 140-141

locating, 128-130

names

- characters*, 115
- spaces*, 115

navigating within, vim text editor, 224-225

opening, vim text editor, 223-224

permissions, 292, 297-298

- default*, 303-305
- finding by*, 302-303, 587
- setting SGID bit*, 300-301

setting sticky bit, 302

special bit, 298-299, 586-587

RPM packages, 69

saving, vim text editor, 226

splitting, 172-173

tar archives, creating, 137-141

testing, 348-349

touching, 120-121

user ownership, changing, 305-306

Filesystem Hierarchy Standard (FHS).

See FHS (Filesystem Hierarchy Standard)

filesystems, 240-241, 255

creating, mkfs command, 260-262

debugging, 268-269

directories, inodes, 258-259

disk space, inodes, 259-260

error checking, 263-266

FHS (Filesystem Hierarchy Standard),
54, 112-113

/usr directory, 114

classifying data, 113-114

commands, 114-136

files, inodes, 256-258

mounting, 269-272

automatically, 273

manually, 272-273

mounts, 16

procf, 8-10

root, 12-14

space utilization, 274

df command, 275-276

disk quotas, 277-282

du command, 274-275

superblocks, 256

tables, 270-272

tuning, 266-267

types, 255-256

unmounting, 273-274

xf, commands, 267-268

filters, 165-174

cut command, 168

expand command, 167-168

head command, 170-171

join command, 168-169

nl command, 166-167

od command, 173-174

paste command, 168-169

sort command, 165-166

tac command, 173

tail command, 171-172

uniq command, 169-170

find command, 130-132, 586-587, 596

finding files by permissions, 302-303

flags setting (xinetd command), 580

flow, email, 484

fmt command, 174-175

fonts, X Window System, 405-406

Foreign Address column (netstat output), 561

format strings, data command, 502

formats, data, transforming, 126-127

formatting commands, 174-175

grep command, 178-184

sed (stream editor) command, 175-176

translating files, 175-176

forwarding email, MTAs (Mail Transfer Agents), 489

free command, 198-201

freezes, X Window System, 409-410

freshening, RPM packages, 74-75

fsck command, 263-266, 269

fstab command, 270

functions, 97

shells, 93, 323-324

fuser command, 596

G

gateway addresses, 537-538

configuring, 550-551

viewing, 550

gdisk command, 250-254

GDM display manager, 408

GECOS field, /etc/passwd file, 424

getent command, 557-559

GIDs (Group IDs), 422-423, 426

entry, /etc/group file, 427

modifying, 433

glob operators, 119

global settings, 94-96

shells, 320

GnuPG keys, 587-590

gpasswd command, 428

GPG, RPM package validation, 71

gpg -gen-key command, 587-589

grace period, disk quotas, 278, 281

graphical package managers, Debian packages, 67

grep command, 178-184

options, 178-179

regular expressions, 185-187

group accounts, 422, 425-427

/etc/group file entries, 427

adding, 431

GIDs, 422-423, 426

limiting, 440

modifying, 432-433

passwords, 427-428

changing, 437-440

primary, 425

removing, 434-435

secondary, 426

security, Shadow Suite, 435-437

UPG (User Private Group), 426-427

GROUP BY clause (SQL), 386-387

group ownership, changing, 306-307

groupadd command, 431

grouping data

SQL (Structured Query Language),
386-387

SQL results, 386-387

groupmod command, 432-433

grpquota command, 277

**GRUB (Grand Unified Boot Loader),
17-18, 31**

GRUB2, 18

command line, 19-20

configuring, 20-21

installing, 18-19

grub.cfg file, 20

grub-mkconfig command, 21

**GUI (graphical user interface), X
Window System, 400-402**

Braille Display, 413

display managers, 408-409

freezes, 409-410

High Contrast/Large Desktop themes,
412

Mouse Keys, 411

onscreen keyboard, 413

remote clients, 413-414

screen reader, 412

Slow/Bounce/Toggle Keys, 411

Sticky/Repeat Keys, 410

Xorg, 402-408

gzip command, 143-144

H

halt command, 42

hanging up processes, 45

hard drives

/dev/sda, 21

disk quotas, 277-282

filesystem mounts, 16

laying out, 11-17

*PIBS (Performance, Integrity, Backup,
Security), 14*

*LVM (Logical Volume Manager),
14-16*

partitions, 11-12

swap, 16-17

tools, 245-255

quotas, 277-282

commands, 277

configuring, 278-280

grace period, 278, 281

hard limit, 277, 280-281

obtaining information, 281-282

soft limit, 277, 280-281

root filesystem, 12-14

space, inodes, 259-260

hard limit, disk quotas, 277, 280-281

hard links, 134-135

hardware

hard drives

/dev/sda, 21

disk quotas, 277-282

filesystem mounts, 16

laying out, 11-17, 14

*LVM (Logical Volume Manager),
14-16*

partitions, 11-12, 16-17, 245-255

quotas, 277-282

root filesystem, 12-14

space, inodes, 259-260

peripherals, 6

compatibility, 7

enumerating, 7-8

integrated, 10-11

procfs, 8-10

viewing list, 8

hardware clock, 500, 503

hwclock command, 503-504

system clock, synchronizing, 504-505

hashall option (bash), 105

.hushlogin file, 425

head command, 170-171

HEADER section (dig query), 558

headers, IPv6, 545

hidden files, 115

**High Contrast theme, X Window
System, 412**

history command, 103-104

history option (bash), 105

history shell, 93

history variables, 104

home directory (FHS), 112

HOME variable, 102-103

/home/ross entry, /etc/passwd file, 424

host command, 554, 557-559

hostname command, 350, 554, 559

hosts, 532

networks, 534

viewing IPs, 555

hwclock command, 503-504

ICMP (Internet Control Message Protocol), 542

idle state, printers, 475

ifconfig command, 554-555

ifdown command, 554

ifup command, 554

IMAP (Internet Message Access Protocol), 488

include subdirectory (/usr directory), 114

inetd command, 576-578

TCP wrappers, 581

init boot option, 32

inodes

directories, 258-259

disk space, 259-260

files, 256-258

input streams

numbering lines, 166-167

splitting, 163

stderr file descriptor, 155-156

stdin file descriptor, 154

INSERT command (SQL), 387**Insert mode, vim text editor, 223****inserting data, SQL (Structured Query Language), 387****install command, 64-65****installation**

Debian packages, dpkg command, 59-60

GRUB2, 18-19

Linux, 3-6

*boot managers, 17-21**laying out hard drive, 11-17**peripherals, 6-8*

remote Debian packages, 64-65

RPM packages, 72-73

YUM packages, 80-83

integers, testing, 349-350**integrated peripherals, 10-11****interfaces**

managing, 545

*configuring default gateway, 550-551**Debian configuration, 548-549**log name configuration, 551-553**Red Hat configuration, 547-548**viewing default gateway, 550**viewing IP information, 545-547*

X Window System, 400-402

*Braille Display, 413**display managers, 408-409**freezes, 409-410**High Contrast/Large Desktop themes, 412**Mouse Keys, 411**onscreen keyboard, 413**remote clients, 413-414**screen reader, 412**Slow/Bounce/Toggle Keys, 411**Sticky/Repeat Keys, 410**Xorg, 402-408***internationalization, 327**

character encoding, 329-330

displaying time, 328

time zones, 327-328

*setting, 328-329***Internet Control Message Protocol (ICMP), 542****Internet Message Access Protocol (IMAP), 488****Internet Printing Protocol (IPP), 468****Internet Protocol (IP), 532**

- interrupt requests (IRQs), CPU/peripheral addressing, 7
- IO port, CPU/peripheral addressing, 7
- IP (Internet Protocol), 532, 541
 - addresses, 533-534
 - viewing information, 545-547
- ip addr show command, 547
- ip command, 554
- ipconfig command, 545-546
- IPP (Internet Printing Protocol), 468
- IPv6, 544-545
 - versus IPv4, 544-545
- IRQs (interrupt requests), CPU/peripheral addressing, 7
- iso9660 filesystem, 255

J

- job control, 205-206
- job control shell, 93
- job scheduling
 - anacron system, 455-456
 - cron system, 447
 - configuring crontabs*, 447-448
 - convenience crontabs*, 454
 - crontab command*, 447-448
 - files*, 452-453
 - finishing crontab*, 450
 - making multiple matches*, 449
 - matching time*, 448-449
 - nicknames extension*, 452
 - output*, 451-452
 - PATH*, 450-451
 - restricting access*, 454-455
 - spelling out month and day names*, 449
 - step values*, 450
 - system crontabs*, 453

- jobs
 - print
 - list*, 477
 - rejecting*, 480-481
 - running ad-hoc, 456-458
 - batch command*, 458-459
- Jobs List (CUPS), 477
- join command, 168-169
- joins, SQL queries
 - advanced, 381-384
 - left versus right, 384
 - subselects, 385-386
 - writing with, 379-380
- journalctl command, 515-516
- journald logging system (systemd), 35
 - configuring, 519-520

K

- KDE desktop, 43
- KDM display manager, 408
- key-value databases, 368-369
- keywords, SQL (Structured Query Language), 372-373
- kill command, 44-45
- killall command, 44-45, 204
- killing processes
 - killall command, 204
 - PID (process ID), 203
 - pkill command, 204-205
- Korn shell, 314
- KVM, 609

L

- language, email, 485-486
- Large Print Desktop theme, X Window System, 412

- last command, 596**
- laying out hard drive, 11-17**
 - PIBS (Performance, Integrity, Backup, Security), 14
- legacy printing tools, CUPS (Common Unix Printing System), 477-480**
- less command, 159**
- lib directory (FHS), 112**
- lib subdirectory (/usr directory), 114**
- libraries**
 - required, 56-57
 - searches, 57
 - shared, 54-56
- LILO boot loader, 31**
- line printer daemon protocol, 467**
- line printer remote (LPR), 467**
- lines**
 - joining, vi text editor, 234
 - numbering, 166-167
- linking files, 133-134**
- Linux**
 - boot process, 30
 - boot loaders, 30-31*
 - common commands, 32-33*
 - phases, 30*
 - systemd, 34-42*
 - SysVinit, 33-34*
 - installing, 3-6
 - boot managers, 17-21*
 - laying out hard drive, 11-17*
 - peripherals, 6-8*
 - Live DVDs, 7
 - open source, 6
 - rebooting from command line, SysVinit, 43
 - shutting down from command line, SysVinit, 42-43
- listing**
 - files, archive, 140-141
 - partitions, 252
- Live DVDs, Linux, 7**
- Loadlin boot loader, 31**
- Local Address column (netstat output), 561**
- local subdirectory (/usr directory), 114**
- locale command, 330-334**
- localization, 327**
 - converting files between encodings, 334
 - fallback locales, 331
 - locale command, 332-333
 - locale contents, 331-332
 - representing locales, 330-331
 - time zones, 327-328
- locate command, 128-130**
- locking console, 213**
- log name configuration, 551-553**
- log_on_failure setting (xinetd command), 580**
- logdump command, 269**
- logger command, 512**
- logging, system**
 - boot events, 45-46
 - journald logging system, configuring, 519-520
 - querying logs, 515-519
 - rotating logs, 520-522
 - syslog, 508-511
 - alternate implementations, 514*
 - systemd, 35, 508-509, 514-515*
 - configuring, 512-514*
 - logger command, 512*
- Logical Volume Manager (LVM), 14-16**
- login files, user accounts, 424-425**

login shell sessions, 95-96, 321
 logind daemon (systemd), 35
 logout, leaving programs running
 after, 209-213
 logs, rotating, 520-522
 loops, 353
 reading from stdin, 356
 sequences, 354-355
 while, 355-356
 lost+found directory (FHS), 112
 lp command, 478
 LPI exams
 certifications and distributions, 610-611
 objectives, 604, 606-608
 preparation, 603-605, 610
 software installation, 611
 studying, 608
 studying don'ts, 609-610
 VMs (virtual machines), 609
 question amount and time, 605
 question types, 611-612, 616
 choose all that apply, 613-614
 choose two/choose three, 613
 fill in the blank, 615-616
 single answer multiple choice, 612
 lpq command, 479
 LPR (line printer remote), 467
 lpr command, 478
 lprm command, 479-480
 LPRng (Next generation of LPR), 467
 lpstat command, 478-479
 ls command, 116-118, 132, 269
 lsblk command, 9
 lscpu command, 9
 lsdev command, 9-10
 lsmod command, 8

lsof command, 596
 lspci command, 9
 lsraid command, 9
 lsscsi command, 9
 lsusb command, 9
 LVM (Logical Volume Manager),
 14-16

M

mail servers, 488-489
 Mail Transfer Agents (MTAs). *See*
 MTAs (Mail Transfer Agents)
 Mail User Agent (MUA), 484-485
 maintenance, CUPS (Common Unix
 Printing System), 474-475
 manually mounting filesystems,
 272-273
 matching time, cron system, 448-449
 math, performing, shell script
 commands, 345-346
 MD5, RPM package validation, 71
 md5sums section (.deb file), 59
 MDAs (Mail Delivery Agents), 488
 media directory (FHS), 112
 mem=xxxxM boot option, 33
 message line, vim text editor, 222
 midnight specifier (at command), 457
 mkdir command, 127-128
 mkfs command, 260-262
 mnt directory (FHS), 112
 Modify Printer command
 (Administration menu), 476
 monitor option (bash), 105
 mount command, 270, 273
 mount units (systemd), 36

mounting filesystems, 16, 269-272

automatically, 273

manually, 272-273

Mouse Keys, X Window System, 411

moving objects, 123-126

MTAs (Mail Transfer Agents), 484-485

aliases, 490

committing changes, 489-490

creating aliases, 489

DNS (domain name system), 487

forwarding email, 489

Linux, 486-487

mail server, 488-489

managing queues, 491-492

MDAs (Mail Delivery Agents), 488

SMTP (Simple Mail Transfer Protocol)

conversation, 485-486

user-defined forwarding, 491

MUA (Mail User Agent), 484-485

**multiple commands, executing,
161-162**

multiple console sessions, 210-211

mv command, 124-126

N

n command, 252-253

**name conventions, RPM packages,
69-70**

**named buffers, vim text editor,
229-230**

names, files

characters, 115

spaces, 115

**navigating files, vim text editor,
224-225**

**navigation commands, FHS
(Filesystem Hierarchy Standard),
115-116**

netstat command, 554, 559-562

network mask, 533, 536

custom, 538-541

default, 536-537

network printing protocols, 467-468

CUPS (Common Unix Printing
System), 468

Administration menu, 476

command line tools, 477

configuration files, 482

configuring, 470-474

configuring of CUPS daemon, 481

daemons, 468

enabling/disabling printers, 480-481

Jobs List, 477

legacy printing tools, 477-480

maintenance, 474-475

*maintenance and administration
pull-downs, 476*

pipeline, 468-470

printer state, 475

rejecting jobs, 480-481

troubleshooting printing, 482-483

**Network Time Protocol (NTP). See
NTP (Network Time Protocol)**

networkd daemon (systemd), 35

networking

address class ranges, 534-535

address classes, 535

broadcast addresses, 538

common ports, 542-544

configuration utilities, 553-554

dbclient command, 556-557

dhcpcd command, 556-557

- dig command*, 557-559
- getent command*, 557-559
- host command*, 557-559
- hostname command*, 559
- ifconfig command*, 555
- netstat command*, 559-562
- ping command*, 562-563
- pump command*, 556-557
- route command*, 555
- tcpdump command*, 565
- traceroute command*, 563-564
- gateway addresses, 533, 537-538
- hosts, 534
- ICMP (Internet Control Message Protocol), 542
- interface management, 545
 - configuring default gateway*, 550-551
 - Debian configuration*, 548-549
 - log name configuration*, 551-553
 - Red Hat configuration*, 547-548
 - viewing default gateway*, 550
 - viewing IP information*, 545-547
- IP (Internet Protocol), 532, 541
 - addresses*, 533-534
- IPv6, 544-545
- network mask, 533, 536
 - custom*, 538-541
 - default*, 536-537
- NPP (Network Printing Protocols), 467-468
 - CUPS (Common Unix Printing System)*, 468-482
- NTP (Network Time Protocol), 504-505
 - configuring ntpd daemon*, 506-507
 - monitoring ntpd daemon*, 507-508
 - pool.ntp.org servers*, 505-506
- TCP (Transmission Control Protocol), 542
- UDP (User Datagram Protocol), 542
- newgrp command**, 428
- Next generation of LPR (LPRng)**, 467
- nfs filesystem**, 256
- nice command**, 207
- nl command**, 166-167
- noclobber option**, 105, 317-318
- noexec option (bash)**, 105
- non-login shell sessions**, 96, 321
- noon specifier (at command)**, 457
- notify option (bash)**, 105
- now + time specifier (at command)**, 457
- nslookup command**, 554
- NTP (Network Time Protocol)**, 504-505
 - configuring ntpd daemon, 506-507
 - monitoring ntpd daemon, 507-508
 - pool.ntp.org servers, 505-506
- ntpd command**, 507-508
- ntpd daemon**
 - configuring, 506-507
 - monitoring, 507-508
- NULL, SQL queries**, 384-385
- numeric mode, permissions**, 294-295

O

objectives, LPI exams, 604, 606-608

objects

- group ownership, changing, 306-307
- moving, 123-126
- permissions, 292
 - default*, 303-305
 - finding files by*, 302-303
 - manipulating*, 294-297

- numeric mode*, 294-295
 - special file*, 297-302
 - symbolic mode*, 296-297
 - trio bits*, 292-294
 - removing, 128
 - user ownership, changing, 305-306
 - od command**, 173-174
 - on demand services**, 576
 - inetd command, 576-578
 - xinetd command, 576-581
 - only_from setting (xinetd command)**, 580
 - onscreen keyboard, X Window System**, 413
 - open command**, 269
 - open source, Linux**, 6
 - opening files, vim text editor**, 223-224
 - operators**
 - glob, 119
 - multiple command, 161-162
 - opt directory (FHS)**, 112
 - output, commands, using from another**, 344-345
 - output streams**
 - processing, 163-164
 - splitting, 163
 - stderr file descriptor, 155-156
 - stdout file descriptor, 154-155
 - tabs, 167-168
 - ownership**
 - group, changing, 306-307
 - user, changing, 305-306
- P**
-
- package management, Debian**, 58-59
 - package managers**, 54
 - packages**, 86
 - Debian, 61-63
 - dependency issues*, 60-61
 - installing*, 59-60
 - reconfiguring*, 61-63
 - remote repositories*, 64
 - removing*, 60
 - remote Debian
 - caches*, 65-66
 - graphical managers*, 67
 - installing*, 64-65
 - removing*, 67
 - updating*, 66-67
 - RPM
 - files*, 69
 - freshening*, 74-75
 - installing*, 72-73
 - name conventions*, 69-70
 - querying*, 77-80
 - removing*, 75-76
 - rpm command*, 70-71
 - upgrading*, 74-75
 - validation*, 71
 - verification*, 73-74
 - YUM
 - configuring*, 84-86
 - finding*, 83-84
 - installing*, 80-83
 - updates*, 83
 - pages**, 199
 - panic=#seconds boot option**, 32
 - Parallels Workstation**, 609
 - parted command**, 254-255
 - partitions**, 11-12, 240-241, 244
 - disk partitioning tools, 245
 - fdisk*, 246-250
 - gdisk*, 250-254
 - parted*, 254-255

- listing, 252
- swap, 16-17
- swap command, 245
- verifying, 253-254
- passwd command, 439, 596**
- passwords**
 - aging, 438-439
 - changing, 437-440
 - encryption, Shadow Suite, 435-437
 - group accounts, 427-428
- paste command, 168-169**
 - vim text editor, 228-229
- path units (systemd), 36**
- PATH variable, 322-323**
 - cron system, 450-451
- paused state, printers, 475**
- Performance, Integrity, Backup, Security (PIBS), 14**
- peripherals, 6**
 - compatibility, 7
 - enumerating, 7-8
 - integrated, 10-11
 - prodfs, 8-10
- permissions, 292, 586-587**
 - directories, setting SGID bit, 301-302
 - files, 297-298
 - default, 303-305*
 - finding by, 302-303, 587*
 - setting SGID bit, 300-301*
 - setting sticky bit, 302*
 - special bit, 298-299, 586-587*
 - manipulating, 294-297
 - numeric mode, 294-295
 - symbolic mode, 296-297
 - trio bits, 292-294
- phases, boot process, 30**
- PIBS (Performance, Integrity, Backup, Security), hard drives, 14**
- PID (process ID), killing processes, 203**
- ping command, 554, 562-563**
- pipeline, CUPS (Common Unix Printing System), 468-470**
- pipes, 159-161**
- pkill command, 204-205**
- pool.ntp.org servers, 505-506**
- POP (Post Office Protocol), 488**
- ports, common, 542-544**
- Post Office Protocol (POP), 488**
- postinst section (.deb file), 59**
- postrm section (.deb file), 59**
- PostScript Printer Description (PPD) file, 473**
- poweroff command, 42**
- PPD (PostScript Printer Description) file, 473**
- PPD files, 482**
- pr command, 174-175**
- prerm section (.deb file), 59**
- primary group accounts, 425**
- print jobs, removing, 479-480**
- print spoolers, 467**
- printers, 467**
 - CUPS (Common Unix Printing System), 468
 - Administration menu, 476*
 - command line tools, 477*
 - configuration files, 482*
 - configuring, 470-474*
 - configuring of CUPS daemon, 481*
 - daemon, 468*
 - enabling/disabling printers, 481*
 - Jobs List, 477*
 - legacy printing tools, 477-480*

- maintenance*, 470-474
 - maintenance and administration*
 - pull-downs*, 476
 - pipeline*, 468-470
 - printer state*, 475
 - rejecting jobs*, 480-481
 - troubleshooting printing*, 482-483
 - print spooler, 467
 - querying status, 479
 - showing status, 478-479
 - state, CUPS (Common Unix Printing System), 475
 - states, 475
 - printers.conf file**, 482
 - printing**, 467
 - command line, 478
 - troubleshooting, 482-483
 - proc directory (FHS)**, 112
 - proc filesystem**, 8-10
 - processes**, 196
 - blocks, 199
 - buffers, 199
 - cached, 199
 - detaching, 212-213
 - interpreting displayed information, 200-201
 - job control, 205-206
 - killing
 - killall command*, 204
 - PID (process ID)*, 203
 - pkill command*, 204-205
 - leaving programs running after logout, 209-213
 - pages, 199
 - properly terminating, 44-45
 - reattaching, 212-213
 - reloading, 45
 - sending signals to, 202-205
 - slabs, 199
 - system uptime, 201-202
 - viewing, 196-198
 - processing, output streams**, 163-164
 - processing state, printers**, 475
 - programs, leaving running after logout**, 209-213
 - Proto column (netstat output)**, 561
 - protocols**
 - ICMP (Internet Control Message Protocol), 542
 - IP (Internet Protocol), 532, 541
 - IPv6, 544-545
 - NTP (Network Time Protocol), 504-508
 - TCP (Transmission Control Protocol), 542
 - UDP (User Datagram Protocol), 542
 - ps command**, 196-198
 - PS1 variable, shells**, 324-325
 - PS2 variable**, 326
 - pstree command**, 197-198
 - pump command**, 554, 556-557
 - pwd command**, 269
- ## Q
-
- queries**
 - dig, 558-559
 - SQL
 - advanced joins*, 381-384
 - cleaning up*, 381
 - left versus right joins*, 384
 - NULL*, 384-385
 - subselects*, 385-386
 - writing with joins*, 379-380

querying

- logs, 515-519
- packages, 61-63
- printer status, 479
- RPM packages, 77-80
 - configuration files*, 79
 - dependencies*, 80

QUESTION section (dig query), 558

queues, mail, managing, 491-492

quit command, 269

quitting vi text editor, 226-227

quota command, 277, 280-282

quotacheck command, 277

quotaon command, 277

quotas (disk)

- commands, 277
- configuring, 278-280
- grace period, 278, 281
- hard limit, 277, 280-281
- obtaining information, 281-282
- soft limit, 277, 280-281

quotoff command, 277

R

RAID (Redundant Array of Independent Disks), 11

rational databases, 369-370

read-only TCP wrappers, 583

real-time clock (RTC). *See* **RTC (real-time clock)**

reattaching processes, 212-213

rebooting from command line,
SysVinit, 43

reconfiguring Debian packages, 61-63

Recv-Q column (netstat output), 561

Red Hat, 50

- interface configuration, 547-548

redirection shell, 93

Redundant Array of Independent Disks (RAID), 11

regular expressions

- grep command, 185-187
- searches, vim text editor, 231-232

ReiserFS, 256

reloading processes, 45

remote clients, X Window System,
413-414

remote Debian packages

- caches, 65-66
- graphical managers, 67
- installing, 64-65
- remote repositories, 64
- removing, 67
- updating, 66-67

renice command, 207-208

Repeat Keys, X Window System, 410

replacing documents, 231

repositories

- remote Debian, 64
- YUM (Yellowdog Updater Modified), 84

Request For Comment (RFC), 534-535

required libraries, 56-57

requires, systemd, 38

results, SQL queries

- limiting, 378
- sorting, 377-378

RFC (Request For Comment),
534-535

ro boot option, 32

root account, access, 574
 su command, 574-575
 sudo command, 575-576

root directory (FHS), 112

root filesystem, hard drives, 12-14

ross,snuffy entry, /etc/group file, 427

Ross Brunson field, /etc/passwd file, 424

ross entry, /etc/passwd file, 424

rotating logs, 520-522

route command, 554-555

routing, IPv6, 545

RPM (Red Hat Package Manager)
 database, 68-69
 packages
 files, 69
 freshening, 74-75
 installing, 72-73
 managing, 68
 name conventions, 69-70
 querying, 77-80
 removing, 75-76
 rpm command, 70-71
 upgrading, 74-75
 validation, 71
 verification, 73-74

rpm command, 70-71

RTC (real-time clock), 503
 hwclock command, 503-504

runlevels, systemd, 37
 changing, 41-42
 determining default, 40-41
 managing, 40
 setting default, 41

running scripts, 343

rw boot option, 32

S

saving files, vim text editor, 226

sbin directory (FHS), 112

sbin subdirectory (/usr directory), 114

scheduling jobs
 anacron system, 455-456
 cron system, 447
 configuring crontabs, 447-448
 convenience crontabs, 454
 crontab command, 447-448
 files, 452-453
 finishing crontab, 450
 making multiple matches, 449
 matching time, 448-449
 nicknames extension, 452
 output, 451-452
 PATH, 450-451
 restricting access, 454-455
 spelling out month and day names, 449
 step values, 450
 system crontabs, 453

schemaless databases, 370-371

scope, variables, 316

scope units (systemd), 36

scp command, 591, 593

screen command, 210-213

screen reader, X Window System, 412

screens, creating windows in, 211-212

scripts, 97
 see also listings
 accepting arguments, 357-358
 case statements, 351
 commands, 344
 combining multiple tests, 351
 performing math, 345-346

- testing files*, 348-349
- testing integers*, 349-350
- testing strings*, 349-350
- use output of another command*, 344-345
- conditions, 346-347
- design, 343
- executing, 94
- interacting with other programs, 356
- loops, 353
 - reading from stdin*, 356
 - sequences*, 354-355
 - while*, 355-356
- returning error code, 357
- running, 343
- sourcing, 94-95
- transferring control to another program, 358
- searches, YUM packages, 83-84**
- searching documents, 231**
- secondary group accounts, 426**
- security**
 - chage command, 596
 - /etc/init.d directory, 596
 - /etc/inittab file, 596
 - /etc/passwd file, 596
 - /etc/shadow file, 596
 - find command, 596
 - fuser command, 596
 - GnuPG keys, 587-590
 - IPv6, 545
 - last command, 596
 - lsof command, 596
 - passwd command, 596
 - permissions, 586-587
 - providing services on demand, 576
 - inetd command*, 576-577
 - root account access, 574
 - su command*, 574-575
 - sudo command*, 575-576
 - services, securing using TCP wrappers, 581-586
 - Shadow Suite, 435-437
 - SSH (Secure Shell), 590-591
 - client utilities*, 592-596
 - components*, 591-592
 - enabling without passwords*, 594-596
 - TCP wrappers
 - configuration*, 583-585
 - hosts.allow/hosts.deny format*, 583
 - read-only*, 583
 - rule options*, 585-586
 - ulimit command, 596
 - usermod command, 596
 - w command, 596
 - who command, 596
- sed command, 176-178**
- SELECT statement, SQL (Structured Query Language), 373-374**
- semicolons, SQL (Structured Query Language), 372**
- sending signals to processes, 202-205**
- Send-Q column (netstat output), 561**
- seq command, 354-355**
- sequences, loops, 354-355**
- server setting (xinetd command), 580**
- service units (systemd), 36**
- services**
 - on demand, 576
 - inetd command*, 576-578
 - xinetd command*, 576-581
 - securing, TCP wrappers, 581-586
- Set Allowed Users command (Administration menu), 476**

Set as Server Default command
(Administration menu), 476

set command, 317-318

Set Default Options command
(Administration menu), 476

severities, syslog, 509

SGID bits

directories, setting, 301-302

files, setting, 300-301

Shadow Suite, 435-437

share subdirectory (/usr directory), 114

shared libraries, 54-56

shell scripting, 338, 342

accepting arguments, 357-358

case statements, 351

conditions, 346-347

interacting with other programs, 356

loops, 353

reading from stdin, 356

sequences, 354-355

while, 355-356

returning error code, 357

script commands, 344

combining multiple tests, 351

performing math, 345-346

testing files, 348-349

testing integers, 349-350

testing strings, 349-350

use output of another command,
344-345

scripts

design, 343

running, 343

transferring control to another
program, 358

shells, 93, 314

bash

history feature, 103-104

setting options, 104-105

env wrapper, 319-320

environment variables, 315

extending, 320

adding dynamic content, 325-326

aliases, 323

creating new users, 326-327

functions, 323-324

global and user settings, 320

login session, 321

non-login session, 321

PATH variable, 322-323

PS1 variable, 324-325

PS2 variable, 326

internationalization, 327

character encoding, 329-330

displaying time, 328

setting time zones, 328-329

time zones, 327-328

localization, 327

converting files between encodings, 334

fallback locales, 331

locale command, 332-333

locale contents, 331-332

representing locales, 330-331

time zones, 327-328

login shell sessions, 95-96

non-login shell sessions, 96

special characters, 99

SSH (Secure Shell), 590-591

client utilities, 592-596

components, 591-592

enabling without passwords, 594-596

- subshells, 318-319
 - T C-shell, 314
 - variables
 - scope*, 316
 - setting and unsetting*, 317-318
 - setting from child*, 316-317
 - working within, 314-320
 - Show All Jobs button (Jobs List)**, 477
 - shutdown command**, 42
 - shutting down from command line,
 - SysVinit, 42
 - SIGHUP signal**, 202
 - SIGINT signal**, 202
 - SIGKILL signal**, 202
 - signals
 - common, 202
 - processes, sending to, 202-205
 - SIGSTOP signal**, 202
 - SIGTERM signal**, 202
 - SIGTSTP signal**, 202
 - Simple Mail Transfer Protocol (SMTP)**, 485-486
 - single boot option, 32
 - skel templates, 429-430
 - slabs, 199
 - slice units (systemd), 36
 - Slow Keys, X Window System**, 411
 - smb filesystem, 256
 - SMTP (Simple Mail Transfer Protocol)**, 485-486
 - snapshot units (systemd), 36
 - socket type setting (xinetd command), 580
 - socket units (systemd), 36
 - soft limit, disk quotas, 277, 280-281
 - software**, 54
 - binaries, 54
 - pieces, 54
 - shared libraries, 54-56
 - sort command**, 165-166
 - sourcing scripts**, 94-95
 - spaces, filenames**, 115
 - special characters, shells**, 99
 - special login files, user accounts**, 424-425
 - special permissions**
 - directories, setting SGID bit, 301-302
 - files, 297-298
 - bit*, 298-299, 586-587
 - setting SGID bit*, 300-301
 - setting sticky bit*, 302
- split command**, 172-173
- split windows, vi text editor**, 234-235
- splitting**
 - files, 172-173
 - streams, 163
- spoolers, print**, 467
- SQL (Structured Query Language)**, 371
 - commands, 372-373
 - comments, 372
 - conditions, multiple, 376-377
 - deleting data, 388
 - grouping data, 386-387
 - inserting data, 387
 - keywords, 372-373
 - queries
 - advanced joins*, 381-384
 - cleaning up*, 381
 - left versus right joins*, 384
 - NULL*, 384-385
 - subselects*, 385-386
 - writing with joins*, 379-380

- results
 - limiting*, 378
 - sorting*, 377-378
- selecting data, 373-374
- semicolons, 372
- SQLite, 371-372
- tables
 - creating*, 388-389
 - multiple*, 378-379
- updating data, 388
- WHERE clause, 374-376
- SQLite, 371-372**
- SSH (Secure Shell), 590-591**
 - client utilities, 592-596
 - components, 591-592
 - enabling without passwords, 594-596
- ssh command, 591-592**
- ssh-add command, 591**
- ssh-agent command, 591**
- sshd command, 591**
- sshd logs, querying, 516**
- stat command, 124**
- State column (netstat output), 561**
- statements**
 - case, 351
 - SELECT, SQL (Structured Query Language), 373-374
- states, printers, 475**
- STATISTICS section (dig query), 558**
- stats command, 269**
- status, printers**
 - querying, 479
 - showing, 478-479
- stderr file descriptor, 155-156**
- stdin file descriptor, 154**
 - reading loops from, 356
- stdout file descriptor, 154-155**
- steams, numbering lines, 166-167**
- step values, cron system, 450**
- sticky bits, files, setting, 302**
- Sticky keys, X Window System, 410**
- stopped state, printers, 475**
- streams**
 - output, processing, 163-164
 - processing output, 163-164
 - sed (stream editor) command, 176-178
 - splitting, 163
 - stderr file descriptor, 155-156
 - stdin file descriptor, 154
 - stdout file descriptor, 154-155
 - tabs, 167-168
- strings, testing, 349-350**
- structuring commands, 98**
- studying for LPI exams, 608**
- su command, 574-575**
- subselects, SQL queries, 385-386**
- subshells, 318-319**
- substitution, commands, 162-163**
- sudo command, 575-576**
- SUID permission set, 300**
- superblocks, filesystems, 256**
- swap command, 245**
- swap partitions, 16-17**
- swap units (systemd), 36**
- symbolic links, 134-135**
- symbolic mode, permissions, 296-297**
- synchronization, clocks, 504-505**
- syntax, test command, 348-349**
- SYSLINUX boot loader, 31**
- syslog**
 - alternate implementations, 514
 - system logging, 508-511

system clock, 500-503

hardware clock, synchronizing, 504-505

system crontabs, 453**system logging**

journald logging system, configuring, 519-520

querying logs, 515-519

rotating logs, 520-522

syslog, 508-511

alternate implementations, 514

systemd, 508-509, 514-515

configuring, 512-514

logger command, 512

system resource allocation, systemd, 36**system time**

clocks, 500

hardware, 500, 503-504

synchronizing, 504-505

system, 500-503

NTP (Network Time Protocol), 504-505

configuring ntpd daemon, 506-507

monitoring ntpd daemon, 507-508

pool.ntp.org servers, 505-506

setting time from command line, 505

system uptime, 201-202**systemctl utility (systemd), 35****systemd, 34-36, 38-42**

component groups, 35

configuring, 512-514

daemons, 35

logging capabilities, 35

runlevels, 37

changing, 41-42

determining default, 40-41

managing system, 40

setting default, 41

service activation, 35

system logging, 508-509

system resource allocation, 36

targets, 37-38

units, 36

Upstart system initialization scheme, 39-40

wants and requires, 38

systemd-analyze utility (systemd), 35**SysVinit, 33-34**

rebooting from command line, 43

shutting down from command line, 42

T**T C-shell, 314****tables**

filesystems, 270-272

SQL (Structured Query Language)

creating, 388-389

multiple, 378-379

tabs, data fields, 167-168**tac command, 173****tail command, 171-172****tar archives, creating, 137-141****tar command, 137-141****target units (systemd), 36****targets, systemd, 37-38****task scheduling, PUT Everything Under Job Scheduling, 451****tasks, PUT Everything Under Jobs, 451****TCP (Transmission Control Protocol), 542**

wrappers

configuration, 583-585

hosts.allow/hosts.deny format, 583

read-only, 583

- rule options, 585-586*
- securing services, 581-586*
- tcpdump command, 554, 565**
- teatime specifier (at command), 457**
- tee command, 163**
- terminating processes, 44-45**
- test command, 348**
 - combining multiple tests, 351
 - syntax, 348-349
 - testing integers, 350
 - testing strings, 349-350
- testing**
 - files, 348-349
 - integers, 349-350
 - strings, 349-350
- tests, combining multiple, 349-350**
- text**
 - changing, 227
 - deleting, 227-228
 - replacing, 227
 - searching in vi, 230
- time**
 - clocks, 500
 - hardware, 500, 503-504*
 - synchronizing, 504-505*
 - system, 500-503*
 - displaying, 328
 - matching, cron system, 448-449
 - NTP (Network Time Protocol), 504-505
 - configuring ntpd daemon, 506-507*
 - monitoring ntpd daemon, 507-508*
 - pool.ntp.org servers, 505-506*
 - setting time from command line, 505*
 - time zones, 327-328
 - setting, 328-329*

- time specifiers, at command, 457**
- time zones, 327-328**
 - setting, 328-329*
- time-of-day specifier (at command), 457**
- timer units (systemd), 36**
- tmp directory (FHS), 112**
- Toggle Keys, X Window System, 411**
- top command, 208-209**
- touch command, 120-121**
- touching files, 120-121**
- tr command, 175-176**
- traceroute command, 554, 563-564**
- Transmission Control Protocol (TCP), 542**
- trio bits, permissions, 292-294**
- troubleshooting, printing, 482-483**
- tune2fs command, 266-267**
- tuning**
 - filesystems, 266-267
 - X Window System, 406-408

U

- udf filesystem, 255**
- UDP (User Datagram Protocol), 542**
- UIDs (User IDs), 422-424**
- ulimit command, 440, 596**
- undelete command, 269**
- undo operators, vim text editor, 225-226**
- uniq command, 169-170**
- units, systemd, 36**
- Unix epoch, 501**
- unmounting, filesystems, 269-274**
- unnamed buffers, vim text editor, 229-230**
- unset command, 317-318**

UPDATE command (SQL), 388

updates, YUM packages, 83

updating

remote Debian packages, 66-67

data, SQL (Structured Query Language), 388

UPG (User Private Group), 426-427

upgrading RPM packages, 74-75

Upstart system initialization scheme, systemd, 39-40

uptime command, 201-202

user accounts, 422

adding, 428-430

entries in /etc/passwd file, 423-424

limiting, 440

modifying, 431-432

passwords, changing, 437-440

removing, 433-434

security, Shadow Suite, 435-437

special login files, 424-425

UIDs (User IDs), 422-424

configuring default gateway, 550-551

Debian configuration, 548-549

log name configuration, 551-553

Red Hat configuration, 547-548

viewing default gateway, 550

viewing IP information, 545-547

User Datagram Protocol (UDP), 542

user interfaces

managing, 545

X Window System, 400-402

Braille Display, 413

display managers, 408-409

freezes, 409-410

High Contrast/Large Desktop themes, 412

Mouse Keys, 411

onscreen keyboard, 413

remote clients, 413-414

screen reader, 412

Slow/Bounce/Toggle Keys, 411

Sticky/Repeat Keys, 410

Xorg, 402-408

user ownership, changing, 305-306

user setting (xinetd command), 580

user settings, 94-96

shells, 320

useradd command, 428-430

defaults, 429

options, 429

skel templates, 429-430

user-defined forwarding, MTAs (Mail Transfer Agents), 491

userdel command, 433-434

usermod command, 431-432, 596

users entry, /etc/group file, 427

usr directory (FHS), 112

/usr directory, FHS (Filesystem Hierarchy Standard), 114

usrquota command, 277

V

validation, RPM packages, 71

var directory (FHS), 112

variables

environment, 101-104, 315

EDITOR, 448

history, 104

PATH, 322-323

PS1, 324-325

PS2, 326

- scope, 316
 - setting and unsetting, 317-318
 - setting from child, 316-317
 - verbose option (bash), 105**
 - verification**
 - partitions, 253-254
 - RPM packages, 73-74
 - vfat filesystem, 255**
 - vga boot option, 32**
 - vi option (bash), 105**
 - vi text editor, 219-218**
 - joining lines, 234
 - options, 232-233
 - quitting, 226-227
 - running external commands, 234
 - split windows, 234-235
 - viewing**
 - default gateways, 550
 - hardware list, 8
 - processes, 196-198
 - vim text editor, 219-222**
 - buffers, 229-230
 - Command mode, 223
 - copy command, 228-229
 - cut command, 228-229
 - editing in vi, 222-223
 - force multipliers, 225
 - Insert mode, 223
 - message line, 222
 - navigating within file, 224-225
 - opening files, 223-224
 - options in vi, 232-233
 - paste command, 228-229
 - regular expression searches, 231-232
 - replacing documents, 231
 - saving files, 226
 - searching documents, 231
 - searching in vi, 230
 - text
 - changing*, 227
 - deleting*, 227-228
 - replacing*, 227
 - undo operators, 225-226
 - VirtualBox, 609**
 - VMs (virtual machines), studying for LPI exams, 609**
 - VMWare, 609**
- ## W
-
- w command, 596**
 - wait setting (xinetd command), 580**
 - wants, systemd, 38**
 - WHERE clause, SQL (Structured Query Language), 374-376**
 - whereis command, 132-133**
 - which command, 132**
 - while loops, 355-356**
 - who command, 596**
 - window managers, X Window System, 401-402**
 - windows, creating in screens, 211-212**
 - wrappers (TCP)**
 - configuration, 583-585
 - hosts.allow/hosts.deny format, 583
 - read-only, 583
 - rule options, 585-586
 - securing services, 581-586

X

x entry

/etc/group file, 427

/etc/passwd file, 424

X Window System, 400-402

Braille Display, 413

desktops, 402

display managers, 408-409

fonts, 405-406

freezes, 409-410

High Contrast/Large Desktop
themes, 412

Mouse Keys, 411

onscreen keyboard, 413

remote clients, 413-414

screen reader, 412

Slow/Bounce/Toggle Keys, 411

Sticky/Repeat Keys, 410

tuning, 406-408

window managers, 401-402

Xorg, 402-408

xargs command, 163-164

XDM display manager, 408

XEN, 609

xfs commands, 267-268

xfs filesystem, 256

xinetd command, 576-581

TCP wrappers, 582-583

Xorg, 402-408

xwininfo command, 407-408

Y

YUM (Yellowdog Updater Modified), 80

packages

configuring, 84-86

finding, 83-84

installing, 80-83

updates, 83

Z

zsh shell, 314

Appendix B

Study Planner

Practice Test	Reading	Task
---------------	---------	------

Element	Task	Goal Date	First Date Completed	Second Date Completed (Optional)	Notes
Introduction	Read Introduction				
1. Installing Linux	Read Foundation Topics				
1. Installing Linux	Review Key Topics				
1. Installing Linux	Define Key Terms				
1. Installing Linux	Answer Review Questions				
Practice Test	Take practice test in study mode using Exam Bank 1 questions for Chapter 1 in practice test software				
2. Boot Process and Runlevels	Read Foundation Topics				
2. Boot Process and Runlevels	Review Key Topics				
2. Boot Process and Runlevels	Define Key Terms				
2. Boot Process and Runlevels	Answer Review Questions				
Practice Test	Take practice test in study mode using Exam Bank 1 questions for Chapter 2 in practice test software				
3. Package Install and Management	Read Foundation Topics				
3. Package Install and Management	Review Key Topics				
3. Package Install and Management	Define Key Terms				
3. Package Install and Management	Answer Review Questions				
Practice Test	Take practice test in study mode using Exam Bank 1 questions for Chapter 3 in practice test software				
4. Basic Command Line Usage	Read Foundation Topics				

4. Basic Command Line Usage	Review Key Topics				
4. Basic Command Line Usage	Define Key Terms				
4. Basic Command Line Usage	Answer Review Questions				
Practice Test	Take practice test in study mode using Exam Bank 1 questions for Chapter 4 in practice test software				
5. File Management	Read Foundation Topics				
5. File Management	Review Key Topics				
5. File Management	Define Key Terms				
5. File Management	Answer Review Questions				
Practice Test	Take practice test in study mode using Exam Bank 1 questions for Chapter 5 in practice test software				
6. Text Processing/Advanced Command Line	Read Foundation Topics				
6. Text Processing/Advanced Command Line	Review Key Topics				
6. Text Processing/Advanced Command Line	Define Key Terms				
6. Text Processing/Advanced Command Line	Answer Review Questions				
Practice Test	Take practice test in study mode using Exam Bank 1 questions for Chapter 6 in practice test software				
7. Process Management	Read Foundation Topics				
7. Process Management	Review Key Topics				
7. Process Management	Define Key Terms				
7. Process Management	Answer Review Questions				
Practice Test	Take practice test in study mode using Exam Bank 1 questions for Chapter 7 in practice test software				
8. Editing Text	Read Foundation Topics				
8. Editing Text	Review Key Topics				
8. Editing Text	Define Key Terms				
8. Editing Text	Answer Review Questions				

Practice Test	Take practice test in study mode using Exam Bank 1 questions for Chapter 8 in practice test software				
9. Partitions and Filesystems	Read Foundation Topics				
9. Partitions and Filesystems	Review Key Topics				
9. Partitions and Filesystems	Define Key Terms				
9. Partitions and Filesystems	Answer Review Questions				
Practice Test	Take practice test in study mode using Exam Bank 1 questions for Chapter 9 in practice test software				
10. Permissions	Read Foundation Topics				
10. Permissions	Review Key Topics				
10. Permissions	Define Key Terms				
10. Permissions	Answer Review Questions				
Practice Test	Take practice test in study mode using Exam Bank 1 questions for Chapter 10 in practice test software				
11. Customizing Shell Environments	Read Foundation Topics				
11. Customizing Shell Environments	Review Key Topics				
11. Customizing Shell Environments	Define Key Terms				
11. Customizing Shell Environments	Answer Review Questions				
Practice Test	Take practice test in study mode using Exam Bank 1 questions for Chapter 11 in practice test software				
12. Shell Scripting	Read Foundation Topics				
12. Shell Scripting	Review Key Topics				
12. Shell Scripting	Define Key Terms				
12. Shell Scripting	Answer Review Questions				
Practice Test	Take practice test in study mode using Exam Bank 1 questions for Chapter 12 in practice test software				
13. Basic SQL Management	Read Foundation Topics				
13. Basic SQL Management	Review Key Topics				

13. Basic SQL Management	Define Key Terms				
13. Basic SQL Management	Answer Review Questions				
Practice Test	Take practice test in study mode using Exam Bank 1 questions for Chapter 13 in practice test software				
14. Configuring User Interfaces and Desktops	Read Foundation Topics				
14. Configuring User Interfaces and Desktops	Review Key Topics				
14. Configuring User Interfaces and Desktops	Define Key Terms				
14. Configuring User Interfaces and Desktops	Answer Review Questions				
Practice Test	Take practice test in study mode using Exam Bank 1 questions for Chapter 14 in practice test software				
15. Managing Users and Groups	Read Foundation Topics				
15. Managing Users and Groups	Review Key Topics				
15. Managing Users and Groups	Define Key Terms				
15. Managing Users and Groups	Answer Review Questions				
Practice Test	Take practice test in study mode using Exam Bank 1 questions for Chapter 15 in practice test software				
16. Schedule and Automate Tasks	Read Foundation Topics				
16. Schedule and Automate Tasks	Review Key Topics				
16. Schedule and Automate Tasks	Define Key Terms				
16. Schedule and Automate Tasks	Answer Review Questions				
Practice Test	Take practice test in study mode using Exam Bank 1 questions for Chapter 16 in practice test software				
17. Configuring Print and Email Services	Read Foundation Topics				
17. Configuring Print and Email Services	Review Key Topics				
17. Configuring Print and Email Services	Define Key Terms				
17. Configuring Print and Email Services	Answer Review Questions				

Practice Test	Take practice test in study mode using Exam Bank 1 questions for Chapter 17 in practice test software				
18. Logging and Time Services	Read Foundation Topics				
18. Logging and Time Services	Review Key Topics				
18. Logging and Time Services	Define Key Terms				
18. Logging and Time Services	Answer Review Questions				
Practice Test	Take practice test in study mode using Exam Bank 1 questions for Chapter 18 in practice test software				
19. Networking Fundamentals	Read Foundation Topics				
19. Networking Fundamentals	Review Key Topics				
19. Networking Fundamentals	Define Key Terms				
19. Networking Fundamentals	Answer Review Questions				
Practice Test	Take practice test in study mode using Exam Bank 1 questions for Chapter 19 in practice test software				
20. Topic 110: Security	Read Foundation Topics				
20. Topic 110: Security	Review Key Topics				
20. Topic 110: Security	Define Key Terms				
20. Topic 110: Security	Answer Review Questions				
Practice Test	Take practice test in study mode using Exam Bank 1 questions for Chapter 20 in practice test software				
21. Final Preparation	Review Exam Essentials for each chapter on the PDF from the DVD				
21. Final Preparation	Review all Key Topics in all chapters				
21. Final Preparation	Take practice test in practice exam mode using Exam Bank #1 questions for all chapters				
21. Final Preparation	Take practice test in practice exam mode using Exam Bank #2 questions for all chapters				

PEARSON IT CERTIFICATION

Browse by Exams ▾

Browse by Technology ▾

Browse by Format

Explore ▾

I'm New Here - Help!

Store

Forums

Safari Books Online

Pearson IT Certification

THE LEADER IN IT CERTIFICATION LEARNING TOOLS

Visit pearsonITcertification.com today to find:

- IT CERTIFICATION EXAM information and guidance for



CompTIA

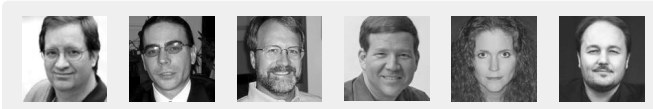
Microsoft

vmware

Pearson is the official publisher of Cisco Press, IBM Press, VMware Press and is a Platinum CompTIA Publishing Partner—CompTIA's highest partnership accreditation

- EXAM TIPS AND TRICKS from Pearson IT Certification's expert authors and industry experts, such as

- *Mark Edward Soper* – CompTIA
- *David Prowse* – CompTIA
- *Wendell Odom* – Cisco
- *Kevin Wallace* – Cisco and CompTIA
- *Shon Harris* – Security
- *Thomas Erl* – SOACP



- SPECIAL OFFERS – pearsonITcertification.com/promotions
- REGISTER your Pearson IT Certification products to access additional online material and receive a coupon to be used on your next purchase

Articles & Chapters



Blogs



Books



Cert Flash Cards Online



eBooks



Mobile Apps



Newsletters



Podcasts



Question of the Day



Rough Cuts



Short Cuts



Software Downloads



Videos

CONNECT WITH PEARSON
IT CERTIFICATION

Be sure to create an account on pearsonITcertification.com and receive members-only offers and benefits





CompTIA Linux+ / LPIC-1 Video Training

From **SANDER VAN VUGT**: Expert author, trainer,
and instructor with more than 20 years of experience



These unique products include

- Live trainer discussions
- CLI presentations
- Live demos
- Whiteboard teaching
- Detailed exam explanations

The *Complete Video Course* also includes

- Interactive hands-on exercises
- Multiple-choice quizzes

livelessons 
video instruction from technology experts

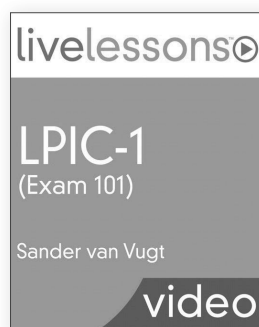
**CompTIA
Linux+ / LPIC-1
Complete
Video Course**
ISBN: 9780789756572

**LPIC-1 (Exam 101)
LiveLessons:
Linux Professional
Institute Certification**

Covers the LPIC-1
(Exam 101) and the
CompTIA Linux+
LX0-103 Exams
ISBN: 9780789754547

**LPIC-1 (Exam 102)
LiveLessons:
Linux Professional
Institute Certification**

Covers the LPIC-1
(Exam 102) and the
CompTIA Linux+
LX0-104 Exams
ISBN: 9780789754776



- These unique video products provide a solid understanding of all topics that candidates need to master to pass the CompTIA Linux+ / LPIC-1 certifications.
- CompTIA Linux+ / LPIC-1 Complete Video Course offers 28 lessons that cover every objective in the CompTIA Linux+ LX0-103 and LX0-104 and the LPIC-1 101 and 102 exams. You also receive practice exam questions, interactive exercises, and CLI simulations so you can practice your skills and knowledge before taking the exams.
- Each course doesn't just offer a preparation for the exams; they also serve as an introduction for people who want to learn basic Linux administration skills.
- Each video provides thorough coverage of command-line skills that work on multiple distributions and prepares anyone who wants to acquire more in-depth knowledge of common Linux administration tasks.
- The course can be used to study for the newest CompTIA Linux+ LX0-103 and LX0-104 exams, the LPIC-1 (Exam 101) and LPIC-1 (Exam 102) certifications, and the SUSE CLA exams.



COMPTIA LINUX+ / LPIC-1 VIDEO TRAINING

Save 40% – Use coupon code LPIC40

PearsonITcertification.com

To receive your 10% off
Exam Voucher, register
your product at:

www.pearsonitcertification.com/register

and follow the instructions.