**Table of Contents**

Title Page

Copyright

Acknowledgments

About the Authors

Introduction

The Exam Objectives

Exam LX0-103 Objectives

Exam LX0-104 Objectives

Assessment Test

Answers to the Assessment Test

Part I: Exam LX0-103

Chapter 1: Exploring Linux Command-Line Tools Understanding Command-Line Basics

Using Streams, Redirection, and Pipes

Processing Text Using Filters

Using Regular Expressions

Summary

Exam Essentials

Review Questions

Chapter 2: Managing Software

Package Concepts

Using RPM

Using Debian Packages

Converting between Package Formats

Package Dependencies and Conflicts

Managing Shared Libraries

Managing Processes

Summary

Exam Essentials

Review Questions

Chapter 3: Configuring Hardware

Configuring the Firmware and Core Hardware Configuring Expansion Cards

Configuring USB Devices

Configuring Hard Disks

Designing a Hard Disk Layout

Creating Partitions and Filesystems

Maintaining Filesystem Health

Mounting and Unmounting Filesystems Summary

Exam Essentials

Review Questions

Chapter 4: Managing Files

Using File Management Commands

Naming Files

Exploring Wildcard Expansion Rules

Understanding the File Commands

Archiving File Commands

Managing Links

Understanding the Directory Commands Managing File Ownership

Controlling Access to Files

Managing Disk Quotas

Locating Files

Summary

Exam Essentials

Review Questions

Chapter 5: Booting Linux and Editing Files

Installing Boot Loaders

Understanding the Boot Process

The Initialization Process

Using the SysV Initialization Process

Using the systemd Initialization Process

Using the Upstart Initialization Process

Editing Files with vi

Summary

Exam Essentials

Review Questions

Part II: Exam LX0-104

Chapter 6: Configuring the X Window System, Localization, and Printing Configuring Basic X Features

Configuring X Fonts

Managing GUI Logins

Using X for Remote Access

X Accessibility

Configuring Localization and Internationalization

Configuring Printing

Summary

Exam Essentials

Review Questions

Chapter 7: Administering the System

Managing Users and Groups

Tuning User and System Environments

Using Log and Journal Files

Maintaining the System Time

Running Jobs in the Future

Summary

Exam Essentials

Review Questions

Chapter 8: Configuring Basic Networking

Understanding TCP/IP Networking

Understanding Network Addressing

Configuring Linux for a Local Network

Diagnosing Network Connections

Summary

Exam Essentials

Review Questions

Chapter 9: Writing Scripts, Configuring Email, and Using Databases Managing the Shell Environment

Writing Scripts

Managing Email

Managing Data with SQL

Summary

Exam Essentials

Review Questions

Chapter 10: Securing Your System

Administering Network Security

Administering Local Security

Configuring SSH

Using GPG

Summary

Exam Essentials

Review Questions

Appendix Answers

Chapter 1: Exploring Linux Command-Line Tools

Chapter 2: Managing Software

Chapter 3: Configuring Hardware

Chapter 4: Managing Files

Chapter 5: Booting Linux and Editing Files

Chapter 6: Configuring the X Window System, Localization, and Printing

Chapter 7: Administering the System

Chapter 8: Configuring Basic Networking

Chapter 9: Writing Scripts, Configuring Email, and Using Databases Chapter 10: Securing Your System

Free Online Learning Environment

End User License Agreement

**List of Illustrations** Figure 2.1

Figure 2.2

Figure 3.1

Figure 3.2

Figure 3.3

Figure 3.4

Figure 3.5

Figure 4.1

Figure 4.2

Figure 5.1

Figure 5.2

Figure 5.3

Figure 6.1

Figure 6.2

Figure 6.3

Figure 6.4

Figure 6.5

Figure 6.6

Figure 7.1

Figure 7.2

Figure 7.3

Figure 8.1

Figure 8.2

Figure 8.3

Figure 8.4

Figure 9.1 Figure 9.2 Figure 10.1 Figure 10.2

**List of Tables** Table 1.1

Table 1.2

Table 1.3

Table 1.4

Table 2.1

Table 2.2

Table 2.3

Table 2.4

Table 2.5

Table 2.6

Table 2.7

Table 3.1

Table 3.2

Table 3.3

Table 3.4

Table 3.5

Table 4.1

Table 4.2

Table 4.3

Table 4.4

Table 4.5

Table 4.6

Table 4.7

Table 4.8

Table 5.1

Table 5.2 Table 8.1 Table 8.2 Table 9.1 Table 9.2 Table 9.3 Table 9.4

**CompTIA Linux+™**

**Powered by Linux Professional Institute Study Guide**

**Exam LX0-103 and Exam LX0-104 Third Edition**

**Christine Bresnahan**

**Richard Blum**

****

Senior Acquisitions Editor: Kenyon Brown Development Editor: Gary Schwartz Technical Editor: Kevin Ryan

Production Editor: Rebecca Anderson Copy Editor: Judy Flynn

Editorial Manager: Mary Beth Wakefield Production Manager: Kathleen Wisor Associate Publisher: Jim Minatel Media Supervising Producer: Richard Graves Book Designers: Judy Fung and Bill Gibson Proofreader: Word One New York

Indexer: Johnna VanHoose

Project Coordinator, Cover: Brent Savage Cover Designer: Wiley

Cover Image: © Getty Images Inc./Jeremy Woodhouse Copyright © 2015 by John Wiley & Sons, Inc., Indianapolis, Indiana Published simultaneously in Canada ISBN: 978-1-119-02121-6

ISBN: 978-1-119-02123-0 (ebk.)

ISBN: 978-1-119-02122-3 (ebk.)

No part of this publication may be reproduced, stored in a retrieval system or transmitted in any form or by any means, electronic, mechanical, photocopying, recording, scanning or otherwise, except as permitted under Sections 107 or 108 of the 1976 United States Copyright Act, without either the prior written permission of the Publisher, or authorization through payment of the appropriate per-copy fee to the Copyright Clearance Center, 222 Rosewood Drive, Danvers, MA 01923, (978) 750-8400, fax (978) 646- 8600. Requests to the Publisher for permission should be addressed to the Permissions Department, John Wiley & Sons, Inc., 111 River Street, Hoboken, NJ 07030, (201) 748-6011, fax (201) 748-6008, or online at www.wiley.com/go/permissions.

Limit of Liability/Disclaimer of Warranty: The publisher and the author make no representations or warranties with respect to the accuracy or completeness of the contents of this work and specifically disclaim all warranties, including without limitation warranties of fitness for a particular purpose. No warranty may be created or extended by sales or promotional materials. The advice and strategies contained herein may not be suitable for every situation. This work is sold with the understanding that the publisher is not engaged in rendering legal, accounting, or other professional services. If professional assistance is required, the services of a competent professional person should be sought. Neither the publisher nor the author shall be liable for damages arising herefrom. The fact that an organization or Web site is referred to in this work as a citation and/or a potential source of further information does not mean that the author or the publisher endorses the information the organization or Web site may provide or recommendations it may make. Further, readers should be aware that Internet Web sites listed in this work may have changed or disappeared between when this work was written and when it is read.

For general information on our other products and services or to obtain technical support, please contact our Customer Care Department within the U.S. at (877) 762-2974, outside the U.S. at (317) 572-3993 or fax (317) 572-4002.

Wiley publishes in a variety of print and electronic formats and by print-on-demand. Some material included with standard print versions of this book may not be included in e-books or in print-on-demand. If this book refers to media such as a CD or DVD that is not included in the version you purchased, you may download this material at http://booksupport.wiley.com. For more information about Wiley products, visit www.wiley.com.

**Library of Congress Control Number:** 2015933954

TRADEMARKS: Wiley, the Wiley logo, and the Sybex logo are trademarks or registered trademarks of John Wiley & Sons, Inc. and/or its affiliates, in the United States and other countries, and may not be used without written permission. CompTIA is a trademark of CompTIA, Inc. All other trademarks are the property of their respective owners. John Wiley & Sons, Inc. is not associated with any product or vendor

property of their respective owners. John Wiley & Sons, Inc. is not associated with any product or vendor mentioned in this book.

**Acknowledgments**

First, all glory and praise go to God, who through His Son, Jesus Christ, makes all things possible, and gives us the gift of eternal life.

Many thanks go to the fantastic team of people at Sybex for their outstanding work on this project. Thanks to Kenyon Brown, the acquisitions editor, for offering us the opportunity to work on this book. Also thanks to Gary Schwartz, the development editor, for keeping things on track and making this book more presentable. Thanks, Gary, for all your hard work and diligence. The technical editor, Kevin E. Ryan, did a wonderful job of double-checking all of the work in the book in addition to making suggestions to improve the content. We would also like to thank Carole Jelen at Waterside Productions, Inc., for arranging this opportunity for us and for helping us out in our writing careers.

Christine would particularly like to thank her husband, Timothy, for his encouragement, patience, and willingness to listen, even when he has no idea what she is talking about.

**About the Authors**

**Richard Blum**, CompTIA Linux+, LPIC-1, has worked in the IT industry for more than 25 years as both a system and network administrator, and he has published numerous Linux and open-source books. Rich is an online instructor for Linux and Web programming courses that are used by colleges and universities across the United States. When he is not being a computer nerd, Rich enjoys spending time with his wife, Barbara, and two daughters, Katie Jane and Jessica.

**Christine Bresnahan,** CompTIA Linux+, LPIC-1, started working with computers more than 25 years ago in the IT industry as a systems administrator. Christine is an Adjunct Professor at Ivy Tech Community College where she teaches Linux certification and Python programming classes. She also writes books and produces instructional resources for the classroom.

**Introduction**

Why should you learn about Linux? It's a fast-growing operating system, and it is inexpensive and flexible. Linux is also a major player in the small and mid size server field, and it's an increasingly viable platform for workstation and desktop use as well. By understanding Linux, you'll increase your standing in the job market. Even if you already know Windows or Mac OS and your employer uses these systems exclusively, understanding Linux will give you an edge when you're looking for a new job or you're looking for a promotion. For instance, this knowledge will help you make an informed decision about if and when you should deploy Linux.

The Computing Technology Industry Association (CompTIA) has developed its Linux+ certification as an introductory certification for people who want to enter careers involving Linux. The exam is meant to certify that an individual has the

skills necessary to install, operate, and troubleshoot a Linux system and is familiar with Linux-specific concepts and basic hardware.

The purpose of this book is to help you pass the Linux+ exams (LX0-103 and LX0-104), updated in 2015. Because these exams cover basic Linux installation, configuration, maintenance, applications, networking, and security, those are the topics that are emphasized in this book. You'll learn enough to get a Linux system up and running and to configure it for many common tasks. Even after you've taken and passed the Linux+ exams, this book should remain a useful reference.

**What Is Linux?**

Linux is a clone of the Unix operating system (OS) that has been popular in academia and many business environments for years. Formerly used exclusively on large mainframes, Unix and Linux can now run on small computers, which are actually far more powerful than the mainframes of just a few years ago. Because of its mainframe heritage, Unix (and hence also Linux) scales well to perform today's demanding scientific, engineering, and network server tasks.

Linux consists of a kernel, which is the core control software, and many libraries and utilities that rely on the kernel to provide features with which users interact. The OS is available in many different distributions, which are collections of a specific kernel with specific support programs.

**Why Become Linux Certified?**

Several good reasons to get your Linux certification exist. There are four major benefits:

**Relevance**

The exams were designed with the needs of Linux professionals in mind. Surveys of Linux administrators were performed to learn what they actually needed to know to do their jobs.

**Quality**

The exams have been extensively tested and validated using psychometric standards. The result is an ability to discriminate between competent administrators and those who must still learn more material.

**Neutrality**

CompTIA is an organization that doesn't itself market any Linux distribution. This fact removes the motivation to create an exam that's designed as a way to market a particular distribution.

**Support**

Major players in the Linux world support the exams.

**How to Become Certified**

The certification is available to anyone who passes the two required exams: LX0-103 and LX0-104 (often referred to as simply 103 and 104). You don't have to work for a particular company. It's not a secret society.

Pearson VUE administers the exam. The exam can be taken at any Pearson VUE testing center. If you pass, you will get a certificate in the mail saying that you have passed.



To register for the exam with Pearson VUE, call (877) 619-2096 or register online at www.vue.com. However you do it, you'll be asked for your name, mailing address, phone number, employer, when and where you want to take the test (that is, which testing center), and your credit card number (arrangement for payment must be made at the time of registration).

**Who Should Buy This Book**

Anybody who wants to pass the certification exams may benefit from this book. This book covers the material that someone new to Linux will need to learn the OS from the beginning, and it continues to provide the knowledge you need up

to a proficiency level sufficient to pass the two exams. You can pick up this book and learn from it even if you've never used Linux before, although you'll find it an easier read if you've at least casually used Linux for a few days. If you're already familiar with Linux, this book can serve as a review and as a refresher course for information with which you might not be completely familiar. In either case, reading this book will help you pass the exams.

This book is written with the assumption that you know at least a little bit about Linux (what it is and possibly a few Linux commands). We also assume that you know some basics about computers in general, such as how to use a keyboard, how to insert a disc into an optical drive, and so on. Chances are that you have used computers in a substantial way in the past—perhaps even Linux, as an ordinary user, or maybe you have used Windows or Mac OS. We do not assume that you have extensive knowledge of Linux system administration, but if you've done some system administration, you can still use this book to fill in gaps in your knowledge.



As a practical matter, you'll need a Linux system with which to practice and learn in a hands-on way. Neither the exams nor this book covers actually installing Linux on a computer from scratch, although some of the prerequisites (such as disk partitioning) are covered. You may need to refer to your distribution's documentation to learn how to accomplish this task. Alternatively, several vendors sell computers with Linux preinstalled.

**How This Book Is Organized**

This book consists of 10 chapters plus supplementary information: an online glossary, this introduction, and the assessment test after the introduction. The chapters are organized as follows:

Chapter 1, “Exploring Linux Command-Line Tools,” covers the basic tools that you need to interact with Linux. These include shells, redirection, pipes, text filters, and regular expressions.

Chapter 2, “Managing Software,” describes the programs that you'll use to manage software. Much of this task is centered around the RPM and Debian package management systems. The chapter also covers handling shared libraries and managing processes (that is, running programs).

Chapter 3, “Configuring Hardware,” focuses on Linux's interactions with the hardware on which it runs. Specific hardware and procedures for using it include the BIOS, expansion cards, USB devices, hard disks, and the partitions and filesystems used on hard disks.

Chapter 4, “Managing Files,” covers the tools used to manage files. This includes commands to manage files, ownership, and permissions as well as Linux's standard directory tree and tools for archiving files.

Chapter 5, “Booting Linux and Editing Files,” explains how Linux boots up and how you can edit files in Linux. Specific topics include the GRUB Legacy and GRUB 2 boot loaders, boot diagnostics, runlevels, and the vi editor.

Chapter 6, “Configuring the X Window System, Localization, and Printing,” describes the Linux GUI and printing subsystems. Topics include X configuration, managing GUI logins, configuring location-specific features, enabling accessibility features, and setting up Linux to use a printer.

Chapter 7, “Administering the System,” describes miscellaneous administrative tasks. These include user and group management, tuning user environments, managing log files, setting the clock, and running jobs in the future.

Chapter 8, “Configuring Basic Networking,” focuses on basic network configuration. Topics include TCP/IP basics, setting up Linux on a TCP/IP network, and network diagnostics.

Chapter 9, “Writing Scripts, Configuring Email, and Using Databases,” covers these miscellaneous topics. Scripts are small programs that administrators often use to help automate common tasks. Email, of course, is an important topic for any computer user, particularly on Linux, which often runs an email server for local or remote use. Linux can run databases that help you store and retrieve information, and these tools can be very important ones on many Linux systems.

Chapter 10, “Securing Your System,” covers security. Specific subjects include network security, local security, and the use of encryption to improve security.

Chapters 1 through 5 cover the LX0-103 exam, while Chapters 6 through 10 cover the LX0-104 exam. These make up Part I and Part II of the book, respectively.

**What's Included in the Book**

We've included several study learning tools throughout the book: **Assessment Test**

At the end of this introduction is an assessment test that you can use to check your readiness for the exam. Take this test before you start reading the book; it will help you determine the areas you might need to brush up on. The answers to the assessment test questions appear on a separate page after the last question of the test. Each answer includes an explanation and a note telling you the chapter in which the material appears.

**Objective Map and Opening List of Objectives**

An objective map shows you where each of the exam objectives is covered in this book. In addition, each chapter opens with a list of the exam objectives it covers. Use these to see exactly where each of the exam topics is covered.

**Exam Essentials**

Each chapter, just after the summary, includes a number of exam essentials. These are the key topics you should take from the chapter in terms of areas to focus on when preparing for the exam.

**Chapter Review Questions**

To test your knowledge as you progress through the book, there are review questions at the end of each chapter. As you finish each chapter, answer the review questions and then check your answers—the correct answers and explanations are in Appendix A. You can go back to reread the section that deals with each question you got wrong to ensure that you answer correctly the next time you're tested on the material.



The review questions, assessment test, and other testing elements included in this book are *not* derived from the actual exam questions, so don't memorize the answers to these questions and assume that doing so will enable you to pass the exam. You should learn the underlying topic, as described in the text of the book. This will help you answer the questions provided with this book *and* pass the exam. Learning the underlying topic is also the approach that will serve you best in the workplace—the ultimate goal of a certification.

To get the most out of this book, you should read each chapter from start to finish and then check your memory and understanding with the end-of-chapter elements. Even if you're already familiar with a topic, you should skim the chapter; Linux is complex enough that there are often multiple ways to accomplish a task, so you may learn something even if you're already competent in an area.

**Interactive Online Learning Environment and Test Bank**

The interactive online learning environment that accompanies the book provides a test bank with study tools to help you prepare for the certification exam—and increase your chances of passing it the first time! The test bank includes the following:

**Sample Tests**

All of the questions in this book are provided, including the Assessment Test, which you'll find at the end of this introduction, and the Chapter Tests that include the Review Questions at the end of each chapter. In addition, there are two Practice Exams. Use these questions to test your knowledge of the study guide material. The online test bank runs on multiple devices.

**Flashcards**

Questions are provided in digital flashcard format (a question followed by a single correct answer). You can use the flashcards to reinforce your learning and provide last-minute test prep before the exam.

**Other Study Tools**

A glossary of key terms from this book and their definitions are available as a fully searchable PDF.



Go to http://sybextestbanks.wiley.com to register and gain access to this interactive online learning environment and test bank with study tools.

**Conventions Used in This Book**

This book uses certain typographic styles in order to help you quickly identify important information and to avoid confusion over the meaning of words such as onscreen prompts. In particular, look for the following styles:

*Italicized text* indicates key terms that are described at length for the first time in a chapter. (Italics are also used for emphasis.)

A monospaced font indicates the contents of configuration files, messages displayed at a text-mode Linux shell prompt, filenames, text-mode command names, and Internet URLs.

*Italicized monospaced text* indicates a variable—information that differs from one system or command run to another, such as the name of a client computer or a process ID number.

**Bold monospaced text** is information that you're to type into the computer, usually at a Linux shell prompt. This text can also be italicized to indicate that you should substitute an appropriate value for your system. (When isolated on their own lines, commands are preceded by non-bold monospaced $ or # command prompts, denoting regular user or system administrator use, respectively.)

In addition to these text conventions, which can apply to individual words or entire paragraphs, a few conventions highlight segments of text:



A note indicates information that's useful or interesting but that's somewhat peripheral to the main text. A note might be relevant to a small number of networks, for instance, or it may refer to an outdated feature.



A tip provides information that can save you time or frustration and that may not be entirely obvious. A tip might describe how to get around a limitation or how to use a feature to perform an unusual task.



Warnings describe potential pitfalls or dangers. If you fail to heed a warning, you may end up spending a lot of time recovering from a bug, or you may even end up restoring your entire system from scratch.

**Sidebar**

A sidebar is like a note but longer. The information in a sidebar is useful, but it doesn't fit into the main flow of the text.



**Real World Scenario**

A real-world scenario is a type of sidebar that describes a task or example that's particularly grounded in the real world. This may be a situation we or somebody we know has encountered, or it may be advice on how to work around problems that are common in real-world, working Linux environments.

**Exercise**

An exercise is a procedure that you should try on your own computer to help you learn about the material in the chapter. Don't limit yourself to the procedures described in the exercises though! Try other commands and procedures to truly learn about Linux.

**The Exam Objectives**

Behind every computer industry exam, you can be sure to find exam objectives —the broad topics in which exam developers want to ensure your competency. The official exam objectives are listed here. (They're also printed at the start of the chapters in which they're covered.)



Exam objectives are subject to change at any time without prior notice and at CompTIA's sole discretion. Please visit CompTIA's website (www.CompTIA.org) for the most current listing of exam objectives.

**Exam LX0-103 Objectives**

Following are the areas in which you must be proficient in order to pass the LX0-103 exam. This exam is broken into four topics (101–104), each of which has three to eight objectives. Each objective has an associated weight, which reflects its importance to the exam as a whole. The four main topics are as follows:

| **Subject Area** |
| --- |
| 101 System Architecture |
| 102 Linux Installation and Package Management |
| 103 GNU and Unix Commands |
| 104 Devices, Linux Filesystems, Filesystem Hierarchy Standard |

**101 System Architecture**

**101.1 Determine and configure hardware settings (Chapter 3)** Enable and disable integrated peripherals

Configure systems with or without external peripherals such as keyboards Differentiate between the various types of mass storage devices Know the differences between coldplug and hotplug devices Determine hardware resources for devices

Tools and utilities to list various hardware information (e.g., lsusb, lspci, etc.)

Tools and utilities to manipulate USB devices

Conceptual understanding of sysfs, udev, hald, dbus

The following is a partial list of the files, terms, and utilities covered in this objective: /sys, /proc, /dev, modprobe, lsmod, lspci, lsusb

**101.2 Boot the system (Chapter 5)**

Provide common commands to the boot loader and options to the kernel at boot time

Demonstrate knowledge of the boot sequence from BIOS to boot completion

Understanding of SysVinit and systemd

Awareness of Upstart

Check boot events in the log file

The following is a partial list of the files, terms and utilities covered in this objective: dmesg, BIOS, bootloader, kernel, init, initramfs, SysVinit, systemd

**101.3 Change runlevels/boot targets and shutdown or reboot system (Chapter 5)**

Set the default runlevel or boot target

Change between runlevels/boot targets, including single user mode Shutdown and reboot from the command line

Alert users before switching runlevels/boot targets or other major system events

Properly terminate processes

The following is a partial list of the files, terms and utilities covered in this objective: *etc*inittab, shutdown, init, *etc*init.d, telinit, systemd, systemctl, *etc*systemd/, *usr*lib/systemd/, wall

**102 Linux Installation and Package Management**

**102.1 Design hard disk layout (Chapter 3)**

Allocate filesystems and swap space to separate partitions or disks Tailor the design to the intended use of the system

Ensure that the /boot partition conforms to the hardware architecture requirements for booting

Knowledge of basic features of LVM

The following is a partial list of the files, terms and utilities covered in this objective: / (root) filesystem, /var filesystem, /home filesystem, /boot filesystem, swap space, mount points, partitions

**102.2 Install a boot manager (Chapter 5)**

Providing alternative boot locations and backup boot options Install and configure a boot loader such as GRUB Legacy

Perform basic configuration changes for GRUB 2

Interact with the boot loader

The following is a partial list of the files, terms, and utilities covered in this objective: /boot/grub/menu.lst, grub.cfg and grub.conf, grub-install, grub-mkconfig, MBR

**102.3 Manage shared libraries (Chapter 2)**

Identify shared libraries

Identify the typical locations of system libraries

Load shared libraries

The following is a partial list of the files, terms and utilities covered in this objective: ldd, ldconfig, *etc*ld.so.conf, LD\_LIBRARY\_PATH

**102.4 Use Debian package management (Chapter 2)** Install, upgrade, and uninstall Debian binary packages

Find packages containing specific files or libraries which may or may not be installed

Obtain package information like version, content, dependencies, package integrity, and installation status (whether or not the package is installed)

The following is a partial list of the files, terms and utilities covered in this objective: *etc*apt/sources.list, dpkg, dpkg-reconfigure, apt-get, apt cache, aptitude

**102.5 Use RPM and YUM package management (Chapter 2)** Install, reinstall, upgrade. and remove packages using RPM and YUM

Obtain information on RPM packages such as version, status, dependencies, integrity, and signatures

Determine what files a package provides, as well as find which package a specific file comes from

The following is a partial list of the files, terms and utilities covered in this objective: rpm, rpm2cpio, *etc*yum.conf, *etc*yum.repos.d/, yum, yumdownloader

**103 GNU and Unix Commands**

**103.1 Work on the command line (Chapter 1)**

Use single shell commands and one-line command sequences to perform basic tasks on the command line

Use and modify the shell environment. including defining, referencing, and exporting environment variables

Use and edit command history

Invoke commands inside and outside the defined path

The following is a partial list of the files, terms, and utilities covered in this objective: bash, echo, env, exec, export, pwd, set, unset, man, uname, history, .bash\_history

**103.2 Process text streams using filters (Chapter 1)**

Send text files and output streams through text utility filters to modify the output using standard Unix commands

The following is a partial list of the files, terms, and utilities covered in this objective: cat, cut, expand, fmt, head, od, join, less, nl, paste, pr, sed, sort, split, tail, tr, unexpand, uniq, wc

**103.3 Perform basic file management (Chapter 4)**

Copy, move, and remove files and directories individually

Copy multiple files and directories recursively

Remove files and directories recursively

Use simple and advanced wildcard specifications in commands Using find to locate and act on files based on type, size, or time Usage of tar, cpio, and dd

The following is a partial list of the files, terms and utilities covered in this objective: cp, find, mkdir, mv, ls, rm, rmdir, touch, tar, cpio, dd, file,

gzip, gunzip, bzip2, xz, file globbing

**103.4 Use streams, pipes, and redirects (Chapter 1)** Redirecting standard input, standard output, and standard error Pipe the output of one command to the input of another command Use the output of one command as arguments to another command Send output to both STDOUT and a file

The following is a partial list of the files, terms and utilities covered in this objective: tee, xargs

**103.5 Create, monitor, and kill processes (Chapter 2)** Run jobs in the foreground and background

Signal a program to continue running after logout

Monitor active processes

Select and sort processes for display

Send signals to processes

The following is a partial list of the files, terms and utilities covered in this objective: &, bg, fg, jobs, kill, nohup, ps, top, free, uptime, pgrep, pkill, killall, screen

**103.6 Modify process execution priorities (Chapter 2)** Know the default priority of a job that is created

Run a program with higher or lower priority than the default Change the priority of a running process

The following is a partial list of the files, terms, and utilities covered in this objective: nice, ps, renice, top

**103.7 Search text files using regular expressions (Chapter 1)** Create simple regular expressions containing several notational elements

Use regular expression tools to perform searches through a filesystem or file content

The following is a partial list of the files, terms and utilities covered in this objective: grep, egrep, fgrep, sed, regex(7)

**103.8 Perform basic file editing operations using vi (Chapter 5)** Navigate a document using vi

Use basic vi modes

Insert, edit, delete, copy, and find text

The following is a partial list of the files, terms and utilities covered in this objective: vi, /, ?, h, j, k, l, i, o, a, c, d, p, y, dd, yy, ZZ, :w!, :q!, :e!

**104 Devices, Linux Filesystems, Filesystem Hierarchy Standard**

**104.1 Create partitions and filesystems (Chapter 3)**

Manage MBR partition tables

Use various mkfs commands to create various filesystems, such as ext2, ext3, ext4, xfs, vfat

Awareness of ReiserFS and Btrfs

Basic knowledge of gdisk and parted with GPT

The following is a partial list of the files, terms, and utilities covered in this objective: fdisk, gdisk, parted, mkfs, mkswap

**104.2 Maintain the integrity of filesystems (Chapter 3)** Verify the integrity of filesystems

Monitor free space and inodes

Repair simple filesystem problems

The following is a partial list of the files, terms, and utilities covered in this objective: du, df, fsck, e2fsck, mke2fs, debugfs, dumpe2fs, tune2fs, xfs tools (such as xfs\_metadump and xfs\_info)

**104.3 Control mounting and unmounting of filesystems (Chapter 3)** Manually mount and unmount filesystems

Configure filesystem mounting on bootup

Configure user mountable removeable filesystems

The following is a partial list of the files, terms, and utilities covered in this objective: *etc*fstab, /media, mount, umount

**104.4 Manage disk quotas (Chapter 4)**

Set up a disk quota for a filesystem

Edit, check, and generate user quota reports

The following is a partial list of the files, terms, and utilities covered in this objective: quota, edquota, repquota, quotaon

**104.5 Manage file permissions and ownership (Chapter 4)**

Manage access permissions on regular and special files as well as directories

Use access modes such as SUID, SGID, and the sticky bit to maintain security

Know how to change the file creation mask

Use the group field to grant file access to group members

The following is a partial list of the files, terms, and utilities covered in this objective: chmod, umask, chown, chgrp

**104.6 Create and change hard and symbolic links (Chapter 4)** Create links

Identify hard and/or soft links

Copying versus linking files

Use links to support system administration tasks

This utility is covered in this objective: ln, ls

**104.7 Find system files and place files in the correct location (Chapter 4)**

Understand the correct locations of files under the FHS

Find files and commands on a Linux system

Know the location and purpose of important files and directories as defined in the FHS

The following is a partial list of the files, terms, and utilities covered in this objective: find, locate, updatedb, whereis, which, type, *etc*updatedb.conf

**Exam LX0-104 Objectives**

The LX0-104 exam comprises six topics (105–110), each of which contains three or four objectives. The six major topics are as follows:

| **Subject Area** |
| --- |
| 105 Shells, Scripting, and Data Management |
| 106 User Interfaces and Desktops |
| 107 Administrative Tasks |
| 108 Essential System Services |
| 109 Networking Fundamentals |
| 110 Security |

**105 Shells, Scripting, and Data Management**

**105.1 Customize and use the shell environment (Chapter 9)**

Set environment variables (e.g., PATH) at login or when spawning a new shell

Write bash functions for frequently used sequences of commands Maintain skeleton directories for new user accounts

Set command search path with the proper directory

The following is a partial list of the files, terms, and utilities covered in this objective: source, *etc*bash.bashrc, *etc*profile, env, export, set, unset, ∼/.bash\_profile, ∼/.bash\_login, ∼/.profile, ∼/.bashrc, ∼/.bash\_logout, function, alias, lists

**105.2 Customize or write simple scripts (Chapter 9)** Use standard sh syntax (loops, tests)

Use command substitution

Test return values for success or failure or other information provided by a command

Perform conditional mailing to the superuser

Correctly select the script interpreter through the shebang (#!) line Manage the location, ownership, execution, and SUID rights of scripts

The following is a partial list of the files, terms, and utilities covered in this objective: for, while, test, if, read, seq, exec

**105.3 SQL data management (Chapter 9)**

Use of basic SQL commands

Perform basic data manipulation

The following is a partial list of the files, terms, and utilities covered in this objective: insert, update, select, delete, from, where, group by, order by, join

**106 User Interfaces and Desktops**

**106.1 Install and configure X11 (Chapter 6)**

Verify that the video card and monitor are supported by an X server Awareness of the X font server

Basic understanding and knowledge of the X Window configuration file

The following is a partial list of the files, terms, and utilities covered in this objective: *etc*X11/xorg.conf, xhost, DISPLAY, xwininfo, xdpyinfo, X

**106.2 Set up a display manager (Chapter 6)**

Basic configuration of LightDM

Turn the display manager on or off

Change the display manager greeting

Awareness of XDM, KDM, and GDM

The following is a partial list of the files, terms, and utilities covered in this objective: lightdm, *etc*lightdm

**106.3 Accessibility (Chapter 6)**

Basic knowledge of keyboard accessibility settings (AccessX) Basic knowledge of visual settings and themes

Basic knowledge of assistive technologies (ATs)

The following is a partial list of the files, terms, and utilities covered in this objective: sticky/repeat keys, slow/bounce/toggle keys, mouse keys, high contrast/large Print Desktop themes, screen reader, Braille display, screen magnifier, onscreen keyboard, Gestures (used at login; for example, gdm), Orca, GOK, emacspeak

**107 Administrative Tasks**

**107.1 Manage user and group accounts and related system files (Chapter 7)**

Add, modify, and remove users and groups

Manage user/group info in password/group databases

Create and manage special-purpose and limited accounts

The following is a partial list of the files, terms, and utilities covered in this objective: *etc*passwd, *etc*shadow, *etc*group, *etc*skel, chage, getent, groupadd, groupdel, groupmod, passwd, useradd, userdel, usermod

**107.2 Automate system administration tasks by scheduling jobs (Chapter 7)**

Manage cron and at jobs

Configure user access to cron and at services

Configure anacron

The following is a partial list of the files, terms, and utilities covered in this objective: *etc*cron.{d,daily,hourly,monthly,weekly}, *etc*at.deny, *etc*at.allow, *etc*crontab, *etc*cron.allow, *etc*cron.deny,

*var*spool/cron/\*, crontab, at, atq, atrm, anacron, *etc*anacrontab

**107.3 Localization and internationalization (Chapter 6)** Configure locale settings and environment variables

Configure time zone settings and environment variables

The following is a partial list of the files, terms, and utilities covered in this objective: *etc*timezone, *etc*localtime, *usr*share/zoneinfo, environment variables (LC\_\*, LC\_ALL, LANG, TZ), *usr*bin/locale, tzselect, tzconfig,

date, iconv, UTF-8, ISO-8859, ASCII, Unicode

**108 Essential System Services**

**108.1 Maintain system time (Chapter 7)**

Set the system date and time

Set the hardware clock to the correct time in UTC

Configure the correct time zone

Basic NTP configuration

Knowledge of using the pool.ntp.org service

Awareness of the ntpq command

The following is a partial list of the files, terms, and utilities covered in this objective: *usr*share/zoneinfo, *etc*timezone, *etc*localtime, *etc*ntp.conf, date, hwclock, ntpd, ntpdate, pool.ntp.org

**108.2 System logging (Chapter 7)**

Configuration of the syslog daemon

Understanding of standard facilities, priorities, and actions

Configuration of logrotate

Awareness of rsyslog and syslog-ng

The following is a partial list of the files, terms, and utilities covered in this objective: syslog.conf, syslogd, klogd, *var*log, logger, logrotate, *etc*logrotate.conf, *etc*logrotate.d/, journalctl,

*etc*system/journal.conf, *var*log/journal/

**108.3 Mail Transfer Agent (MTA) basics (Chapter 9)**

Create email aliases

Configure email forwarding

Knowledge of commonly available MTA programs (postfix, sendmail, qmail, exim) (no configuration)

The following is a partial list of the files, terms, and utilities covered in this objective: ∼/.forward, sendmail emulation layer commands, newaliases,

mail, mailq, postfix, sendmail, exim, qmail

**108.4 Manage printers and printing (Chapter 6)**

Basic CUPS configuration (for local and remote printers)

Manage user print queues

Troubleshoot general printing problems

Add and remove jobs from configured printer queues

The following is a partial list of the files, terms, and utilities covered in this objective: CUPS configuration files, tools and utilities; *etc*cups; lpd legacy interface (lpr, lprm, lpq)

**109 Networking Fundamentals**

**109.1 Fundamentals of Internet protocols (Chapter 8)** Demonstrate an understanding of network masks and CIDR notation

Knowledge of the differences between private and public “dotted quad” IP addresses

Knowledge about common TCP and UDP ports (20, 21, 22, 23, 25, 53, 80, 110, 123, 139, 143, 161, 162, 389, 443, 465, 514, 636, 993, 995)

Knowledge about the differences and major features of UDP, TCP, and ICMP

Knowledge of the major differences between IPv4 and IPV6

Knowledge of the basic features of IPv6

The following is a partial list of the files, terms, and utilities covered in this objective: *etc*services, IPv4, IPv6, subnetting, TCP, UDP, ICMP

**109.2 Basic network configuration (Chapter 8)**

Manually and automatically configure network interfaces

Basic TCP/IP host configuration

Setting a default route

The following is a partial list of the files, terms, and utilities covered in this objective: *etc*hostname, *etc*hosts, *etc*nsswitch.conf, ifconfig, ifup,

ifdown, ip, route, ping

**109.3 Basic network troubleshooting (Chapter 8)**

Manually and automatically configure network interfaces and routing tables to include adding, starting, stopping, restarting, deleting, or reconfiguring network interfaces

Change, view, or configure the routing table and correct an improperly set default route manually

Debug problems associated with the network configuration

The following is a partial list of the files, terms, and utilities covered in this objective: ifconfig, ip, ifup, ifdown, route, host, hostname, dig, netstat, ping, ping6, traceroute, traceroute6, tracepath, tracepath6, netcat

**109.4 Configure client-side DNS (Chapter 8)**

Query remote DNS servers

Configure local name resolution and use remote DNS servers Modify the order in which name resolution is done

The following is a partial list of the files, terms, and utilities covered in this objective: *etc*hosts, *etc*resolv.conf, *etc*nsswitch.conf, host, dig, getent

**110 Security**

**110.1 Perform security administration tasks (Chapter 10)** Audit a system to find files with the SUID/SGID bit set

Set or change user passwords and password aging information Be able to use nmap and netstat to discover open ports on a system Set up limits on user logins, processes, and memory usage

Determine which users have logged in to the system or are currently logged in

Basic sudo configuration and usage

The following is a partial list of the files, terms, and utilities covered in this objective: find, passwd, lsof, nmap, chage, netstat, sudo, *etc*sudoers, su, usermod, ulimit, who, w, last

**110.2 Set up host security (Chapter 10)**

Awareness of shadow passwords and how they work

Turn off network services not in use

Understand the role of TCP wrappers

The following is a partial list of the files, terms, and utilities covered in this objective: *etc*nologin, *etc*passwd, *etc*shadow, *etc*xinetd.d/\*, *etc*xinetd.conf, *etc*inetd.d/\*, *etc*inetd.conf, *etc*inittab, *etc*init.d/\*, *etc*hosts.allow, *etc*hosts.deny

**110.3 Securing data with encryption (Chapter 10)**

Perform basic OpenSSH 2 client configuration and usage

Understand the role of OpenSSH 2 server host keys

Perform basic GnuPG configuration and usage

Understand SSH port tunnels (including X11 tunnels)

The following is a partial list of the files, terms, and utilities covered in this objective: ssh, ssh-keygen, ssh-agent, ssh-add, ∼/.ssh/id\_rsa and id\_rsa.pub, ∼/.ssh/id\_dsa and id\_dsa.pub, *etc*ssh/ssh\_host\_rsa\_key and ssh\_host\_rsa\_key.pub, *etc*ssh/ssh\_host\_dsa\_key and ssh\_host\_dsa\_key.pub, ∼/.ssh/authorized\_keys, *etc*ssh\_known\_hosts, gpg, ∼/.gnupg/\*

**Assessment Test**

**1.** Which section in the X server configuration file defines the combination of monitors and video cards that you're using on your Linux system?

**A.** Monitor

**B.** Screen

**C.** Modeline

**D.** Device

**E.** Module

**2.** How can you tell whether your system is using inetd or xinetd as a super server? (Select two.)

**A.** Type **ps ax | grep inetd**, and examine the output for signs of inetd or xinetd.

**B.** Type **superserver** to see a report on which super server is running.

**C.** Look for the *etc*inetd.conf file or *etc*xinetd.d subdirectory, which are signs of inetd or xinetd, respectively.

**D.** Examine the *etc*inittab file to see which super server is launched by init, which is responsible for this task.

**E.** Type **netstat -a | grep inet** and examine the output for signs of inetd or xinetd.

**3.** How does the lpc utility for CUPS differ from its counterpart in BSD LPD and LPRng?

**A.** The lpc utility is unique to CUPS; it doesn't ship with BSD LPD or LPRng. **B.** CUPS doesn't ship with an lpc command, but BSD LPD and LPRng do.

**C.** CUPS's lpc is much more complex than its counterpart in BSD LPD and LPRng.

**D.** CUPS's lpc is much simpler than its counterpart in BSD LPD and LPRng. **E.** The lpc utility is identical in all three of these printing systems. **4.** What file would you edit to restrict the number of simultaneous logins a user

can employ?

**A.** *etc*pam.d/login-limits

**B.** *etc*bashrc

**C.** *etc*security/limits.conf

**D.** *etc*inittab

**E.** *etc*passwd

**5.** Which of the following are required when configuring a computer to use a static IP address? (Select two.)

**A.** The IP address of the DHCP server

**B.** The hostname of the NBNS server

**C.** The computer's IP address

**D.** The network mask

**E.** The IP address of the NTP server

**6.** What does the following command accomplish?

$ **wc report.txt | tee wc**

**A.** It launches the wc editor on both the report.txt and wc.txt files; each file opens in its own window.

**B.** It displays a count of the windows in which the report.txt file is displayed and shows that information in a new window called wc.

**C.** It creates a count of newlines, words, and bytes in the report.txt file and then displays a count of these statistics about the report it just generated.

**D.** It cleans up any memory leaks associated with the tee program's use of the report.txt file.

**E.** It displays a count of newlines, words, and bytes in the report.txt file and copies that output to the wc file.

**7.** Which of the following characters defines the *end* of an OS or kernel definition in *boot*grub/grub.cfg?

**A.** ;

**B.** )

**C.** }

**D.** \*/

**E.** None of the above; the definition ends with the title line beginning the next entry.

**8.** What does the number 703 represent in the following *etc*passwd entry? george:x:703:100:George Brown:/home/george:/bin/tcsh **A.** The account's human ID (HID) number

**B.** The account's process ID (PID) number

**C.** The account's group ID (GID) number

**D.** The account's globally unique ID (GUID) number

**E.** The account's user ID (UID) number

**9.** What does the grep command accomplish?

**A.** It creates a pipeline between two programs.

**B.** It searches files' contents for a pattern.

**C.** It concatenates two or more files.

**D.** It displays the last several lines of a file.

**E.** It locates files on the hard disk.

**10.** Which of the following are journaling filesystems for Linux? (Select three.) **A.** vfat

**B.** ReiserFS

**C.** Ext2fs

**D.** Ext3fs

**E.** XFS

**11.** You've configured your computer to use SMTP and IMAP via a tunneled SSH connection to your ISP's email server for improved security. Why might you still want to use GPG encryption for your emails on top of the encryption provided by SSH?

**A.** The SSH tunnel reaches only as far as the first email server; GPG encrypts

data on all of the computers all the way to or from your email correspondents.

**B.** SSH encryption is notoriously poor for email, although it's perfectly adequate for login sessions; thus, adding GPG encryption improves security.

**C.** SSH doesn't encrypt the headers of the email messages; GPG encrypts the headers to keep snoopers from learning your correspondents' identities.

**D.** Using GPG guarantees that your email messages won't contain unwanted viruses or worms that might infect your correspondents' computers.

**E.** Configured in this way, SSH will encrypt the email headers and bodies but not any attachments to your email.

**12.** Which of the following ports are commonly used to retrieve email from an email server computer? (Select two.)

**A.** 110

**B.** 119

**C.** 139

**D.** 143

**E.** 443

**13.** You're experiencing sporadic problems with a Secure Shell (SSH) login server—sometimes users can log in and sometimes they can't. What might you try immediately after a failure to help diagnose this problem?

**A.** On the server computer, type **http://localhost:631** into a web browser to access the SSH configuration page and check its error subpage for error messages.

**B.** Type **diagnose sshd** to run a diagnostic on the SSH server daemon (sshd). **C.** Type **tail *var*log/messages** to look for error messages from the server. **D.** Examine the *dev*ssh device file to look for error messages from the server. **E.** On the server computer, type **sshd** to view SSH's diagnostic messages. **14.** What is the function of the ∼/.profile file?

**A.** It's the user configuration file for the ProFTP server.

**B.** It's one of a user's bash startup scripts.

**C.** It's the user configuration file for the ProFile file manager. **D.** Its presence tells tcsh to ignore file modes.

**E.** It holds the user's encrypted password.

**15.** You want your computer to remind you to get your car inspected in two years. What is the best way to do this among the specified options?

**A.** Create a program that repeatedly checks the time and, when two years have passed, displays a message to get your car inspected.

**B.** Type **cal day month year**, where *day*, *month*, and *year* specify the date of the future inspection, to have Linux run a program that you then specify on that date.

**C.** Create a cron job that runs hourly. This job should check the date and, when the correct date comes up, use mail to notify you of the need for a car inspection.

**D.** Use the NTP GUI calendar program to create an alarm for the specified date. The program will then display the message you enter at the specified date and time.

**E.** Type **at date**, where *date* is a date specification. You can then specify a command, such as mail with appropriate options, to notify you of the need to get your car inspected.

**16.** How would you configure a computer to use the computer whose IP address is 172.24.21.1 as a gateway for all network traffic that's not otherwise configured?

**A. gateway default 172.24.21.1**

**B. gateway 172.24.21.1**

**C. route gateway 172.24.21.1**

**D. route add default gw 172.24.21.1**

**E. gw 172.24.21.1**

**17.** What software can you use to drive a Braille display device? (Select two.) **A.** Emacspeak

**B.** BRLTTY

**C.** A 2.6.26 or later kernel

**D.** GOK

**E.** A framebuffer driver

**18.** Which is true of source RPM packages?

**A.** They consist of three files: an original source tarball, a patch file of changes, and a PGP signature indicating the authenticity of the package.

**B.** They require programming knowledge to rebuild.

**C.** They can sometimes be used to work around dependency problems with a binary package.

**D.** They are necessary to compile software for RPM-based distributions. **E.** They always contain software that's licensed under terms of the GPL.

**19.** Which utility should you use by itself to rename the file pumpkin.txt to lantern.txt?

**A.** dd

**B.** rm

**C.** cp

**D.** mv

**E.** ln

**20.** You want to run a lengthy scientific simulation program, called simbigbang, which doesn't require any user interaction; the program operates solely on disk files. If you don't want to tie up the shell from which you run the program, what should you type to run simbigbang in the background?

**A. start simbigbang**

**B. simbigbang &**

**C. bg simbigbang**

**D. background simbigbang**

**E. nice simbigbang**

**21.** Which of the following commands will install an RPM package file called theprogram-1.2.3-4.i386.rpm on a computer? (Select two.)

**A. rpm -Uvh theprogram-1.2.3-4.i386.rpm**

**B. rpm -i theprogram-1.2.3-4.i386.rpm**

**C. rpm -U theprogram**

**D. rpm -e theprogram-1.2.3-4.i386.rpm**

**E. rpm -Vp theprogram-1.2.3-4.i386.rpm**

**22.** What tool can diagnose and fix many common Linux filesystem problems? **A.** mkfs

**B.** fsck

**C.** chkdsk

**D.** scandisk

**E.** fdisk

**23.** You've just installed MySQL, and you intend to use it to store information about the animals in a zoo, from the anteaters to the zebras. What command are you likely to use first, once you start MySQL?

**A. CREATE DATABASE animals;**

**B. USE animals;**

**C. CREATE TABLE animals;**

**D. INSERT INTO animals;**

**E. UPDATE animals;**

**24.** Which of the following commands displays help on *topic*, when typed in a Linux shell? (Select two.)

**A. manual *topic***

**B. man *topic***

**C. ? *topic***

**D. info *topic***

**E. hint *topic***

**25.** A computer's hardware clock keeps track of the time while the computer is powered off. In what formats may this time be stored on an *x*86 Linux system? (Select two.)

**A.** Coordinated Universal Time (UTC)

**B.** Internet Time

**C.** Local time

**D.** 12-hour time

**E.** Mars time

**26.** You want to know what kernel modules are currently loaded. What command would you type to learn this information?

**A. insmod**

**B. depmod**

**C. modprobe**

**D. lsmod**

**E. modinfo**

**27.** You want to enable all members of the music group to read the instruments.txt file, which currently has 0640 (-rw-r-----) permissions, ownership by root, and group ownership by root. How might you accomplish this goal? (Select two.)

**A.** Type **chown music instruments.txt** in the file's directory. **B.** Type **chgrp music instruments.txt** in the file's directory. **C.** Type **chgroup music instruments.txt** in the file's directory. **D.** Type **chmod 0600 instruments.txt** in the file's directory. **E.** Type **chown :music instruments.txt** in the file's directory.

**28.** You want to create a link to the *usr*local/bin directory in another location. Which of the following statements is true?

**A.** You can do this only if *usr*local/bin is on a journaling filesystem. **B.** You must own *usr*local/bin to create the link.

**C.** You can create the link only if the link's location is on the same filesystem as the original directory.

**D.** Only the system administrator can do this.

**E.** The link will probably have to be a symbolic link.

**29.** Which of the following, when typed in vi's command mode, saves a file and quits the program? (Select two.)

**A. :rq**

**B. :wq**

**C. :re**

**D. :we**

**E. ZZ**

**30.** A user's home directory includes a file called ∼/.forward that consists of one line: |∼/junkme. What is the effect of this configuration?

**A.** The user's incoming mail is forwarded to the junkme user on the same system.

**B.** The user's incoming mail is stored in the ∼/junkme file.

**C.** The user's incoming mail is sent through the ∼/junkme program file. **D.** The user's incoming mail is flagged as spam and deleted.

**E.** The user's incoming mail is forwarded to the same user on the junkme computer.

**Answers to the Assessment Test**

**1.** B. The Monitor section defines the monitor options and settings but doesn't combine it with the video card, so option A is incorrect. The Modeline line defines the available video modes in the Monitor section, but it doesn't define video cards, so option C is incorrect. Option D, the Device section, is also incorrect; it defines the video card but doesn't match it with a monitor on the system. Option E is incorrect because the Module section defines which X server modules (or drivers) are loaded but it doesn't match monitors and video cards. Option B, the Screen section, tells the X server about the combination of video cards and monitors that you're using, so it's the correct answer. For more information, see Chapter 6, “Configuring the X Window System, Localization, and Printing.”

**2.** A, C. Examining a process listing (obtained from ps) for signs of the super server is the most reliable way to determine which one is actually running, so option A is correct. The presence of the super server's configuration file or files (as in option C) is also a good diagnostic, although some older

systems that have been upgraded may have both sets of configuration files. There is no standard superserver utility to report on which one is used, so option B is incorrect. Most distributions launch the super server through a

SysV startup script; the *etc*inittab file isn't directly involved in this process, so examining it would be pointless, and option D is incorrect. Although the output of **netstat -ap**, when typed as root, will include an indication of any instance of inetd or xinetd that's listening for connections, option E omits the critical -p option, which causes the program to display process names. Thus, option E is incorrect. For more information, see Chapter 10, “Securing Your System.”

**3.** D. The lpc utility is used to start, stop, change the priority of, and otherwise control jobs in a print queue. CUPS ships with an lpc utility, but it's quite rudimentary compared to the lpc utilities of BSD LPD and LPRng. Instead, CUPS relies on its Web-based interface to provide the ability to control print jobs. Thus, option D is correct, and the remaining options must logically all be incorrect. For more information, see Chapter 6.

**4.** C. The *etc*security/limits.conf file defines various limits on user resources, including the number of simultaneous logins individual users are permitted. Thus, option C is correct. The *etc*pam.d/login-limits file

(option A) is fictitious, although login limits do rely on the pam\_limits module to the Pluggable Authentication System (PAM). The *etc*bashrc file (option B) is a global bash startup script file, but it's not normally used to impose login limits. The *etc*inittab file (option D) is a key Linux startup file, but it doesn't have any direct bearing on imposing login limits. The *etc*passwd file (option E) defines many key account features, but login limits are not among these. For more information, see Chapter 10.

**5.** C, D. The computer's IP address (option C) and network mask (aka subnet mask or netmask; option D) are the most critical components in TCIP/IP network configuration. (Additional information that you may need to provide on many networks includes the IP addresses of one to three DNS servers, the hostname or IP address of a router, and the computer's hostname.) You shouldn't need the IP address of a Dynamic Host Configuration Protocol (DHCP) server (option A)—and if a DHCP server is present, chances are you should be using DHCP rather than static IP address assignment. A NetBIOS Name Service (NBNS) server (option B) converts between names and IP addresses on NetBIOS networks. The hostname of such a computer isn't likely to be a critical configuration element, although you may need to provide this information to Samba for some operations to function correctly when sharing files. A Network Time Protocol (NTP) server (option E) helps you maintain system time on all of your computers, but this isn't required for basic network configuration. For more information, see Chapter 8, “Configuring Basic Networking.”

**6.** E. The wc command displays a count of newlines, words, and bytes in the specified file (report.txt). Piping this data through tee causes a copy of the output to be stored in the new file (wc in this example—you shouldn't run this command in the same directory as the wc executable file!). Thus, option E is correct. Contrary to option A, wc is not an editor, and the remaining syntax wouldn't cause two files to open in separate windows even if wc were an editor. Contrary to option B, wc doesn't count windows or open a new window. Option C describes the effect of **wc report | wc**— that is, it overlooks the tee command. Contrary to option D, wc has nothing to do with cleaning up memory leaks, and tee doesn't directly use the report.txt file. For more information, see Chapter 1, “Exploring Linux Command-Line Tools.”

**7.** C. The grub.cfg filename indicates a GRUB 2 configuration file. In such files, each OS or kernel stanza begins with a menuentry line and an open

curly brace ({) and ends with a close curly brace (}). Thus, option C is correct. Some configuration files and programming languages use semicolons (;) at the end of most lines, but this isn't true of GRUB 2, so option A is incorrect. Although close parentheses ()) are used to terminate some types of options in some configuration files, including disk identifiers in GRUB 2's configuration file, they aren't used to terminate whole OS or kernel definitions in this file, so option B is incorrect. The string \*/ terminates comments in C program files but isn't commonly used in GRUB 2 configuration files, so option D is incorrect. Option E would be correct if the question had asked about a GRUB Legacy configuration file (menu.lst or grub.conf), but the question specifies a GRUB 2 configuration file (grub.cfg); the two boot loaders terminate their OS/kernel stanzas differently, so option E is incorrect. For more information, see Chapter 5, “Booting Linux and Editing Files.”

**8.** E. The third field of *etc*passwd entries holds the UID number for the account, so option E is correct. Linux doesn't use any standard identifier called a human ID (HID; option A), although the acronym HID stands for human interface device, a class of USB devices. Accounts don't have PID numbers (option B); those belong to running processes. The account's GID number (option C) is stored in the fourth field of *etc*passwd—100 in this example. Linux accounts don't use globally unique ID (GUID) numbers, so option D is incorrect. For more information, see Chapter 7, “Administering the System.”

**9.** B. The grep command scans files to find those that contain a specified string or pattern, as described by option B. In the case of text files, grep displays the matching line or lines; for binary files, it reports that the file matches the pattern. The method of creating a pipeline (option A) involves separating two commands with a vertical bar (|). The grep command can be used in a pipeline, but it doesn't create one. The command that concatenates files (option C) is cat, and the command that displays the last several lines of a file (option D) is tail. Several commands, such as find, locate, and whereis locate files (option E), but grep is not among them. For more information, see Chapter 1.

**10.** B, D, E. ReiserFS (option B) was written from scratch for Linux. The Third Extended Filesystem (ext3fs; option D) is a journaling filesystem based on the older non-journaling Second Extended Filesystem (ext2fs; option C). The Extents Filesystem (XFS; option E) is a journaling

filesystem written by SGI for Irix and later ported to Linux. The Virtual File Allocation Table (vfat; option A) is a non-journaling filesystem designed by Microsoft for Windows. For more information, see Chapter 3, “Configuring Hardware.”

**11.** A. Option A correctly describes the features of SSH and GPG in this context. Option B is incorrect because SSH should do a fine job of encrypting your email so that it can't be decoded between your system and your ISP's email server. Option C has it backward; email transferred via SSH will be completely encrypted, including both headers and body. GPG doesn't encrypt headers, just message bodies. Option D is incorrect because GPG isn't a virus scanner, just an encryption tool. Option E is incorrect because the SSH tunnel will encrypt everything in the SMTP transfer, including email attachments. For more information, see Chapter 10.

**12.** A, D. Port 110 (option A) is assigned to the Post Office Protocol (POP), and port 143 (option D) is assigned to the Internet Message Access Protocol (IMAP), both of which may be used to retrieve email messages from an email server system. Port 119 (option B) is assigned to the Network News Transfer Protocol (NNTP), port 139 (option C) is assigned to the Server Message Block/Common Internet File System (SMB/CIFS) protocol, and port 443 (option E) is assigned to the Hypertext Transfer Protocol with SSL encryption (HTTPS), none of which is commonly used for email retrieval. For more information, see Chapter 8.

**13.** C. Log files, such as *var*log/messages and sometimes others in *var*log, often contain useful information concerning server errors. The tail program displays the last few lines of a file, so using it to examine log files immediately after a problem occurs can be a useful diagnostic procedure. Option C correctly combines these features. The http://localhost:631 URL of option A accesses the Common Unix Printing System (CUPS) configuration utility, which has nothing to do with SSH. There is no standard diagnose utility (option B) to help diagnose server problems, and there is no standard *dev*ssh file (option D). The **sshd** program is the SSH server itself, so option E will simply launch the server. For more information, see Chapter 5.

**14.** B. The ∼./profile file is one of several bash startup scripts, as stated in option B. It has nothing to do with the ProFTP server (option A) or the tcsh shell (option D). The ProFile file manager mentioned in option C is fictitious. Users' encrypted passwords (option E) are usually stored in

*etc*shadow. For more information, see Chapter 9, “Writing Scripts, Configuring Email, and Using Databases.”

**15.** E. The at utility was created to run programs at one specified point in the future. Thus, option E will accomplish the stated goal. Options A and C might also work, but neither is the *best* way to accomplish this goal. Option A will tie up CPU time, and if the program crashes or the system is shut down during the intervening two years, the message will never be displayed. Option C would be more reliable, but it adds unnecessary complexity to your hourly cron job schedule. The cal program displays a text-mode calendar, enabling you to identify the days of a week for a given month; it doesn't schedule future jobs, as option B suggests. A GUI calendar program, as specified in option D, might work, but NTP is the Network Time Protocol, a protocol and like-named program for synchronizing clocks across a network. Thus, NTP isn't the tool for the job, and option D is incorrect. For more information, see Chapter 7.

**16.** D. Option D provides the correct command to add 172.24.21.1 as the default gateway. Options A and B both use the fictitious gateway command, which doesn't exist and therefore won't work unless you create a script of this name. Option C uses the correct route command, but there is no gateway option to route; you must use add default gw, as in option D. There is no standard gw command, so option E is incorrect. For more information, see Chapter 8.

**17.** B, C. The BRLTTY package is an add-on daemon for handling a Braille display device, and some features for using these devices have been added to the 2.6.26 kernel, so options B and C are correct. Emacspeak (option A) is speech-synthesis software; it can be used to “speak” a text display to a user, but it doesn't interface with Braille displays. GOK (option D) is an onscreen keyboard, not a Braille display tool. Framebuffer drivers (option E) are kernel drivers for managing conventional video cards; they aren't used to drive Braille displays. For more information, see Chapter 6.

**18.** C. Some dependencies result from dynamically linking binaries to libraries at compile time, and so they can be overcome by recompiling the software from a source RPM, so option C is correct. Option A describes Debian source packages, not RPM packages. Recompiling a source RPM requires only issuing an appropriate command, although you must also have appropriate compilers and libraries installed. Thus, option B is overly pessimistic. Source tarballs can also be used to compile software for RPM

systems, although this results in none of RPM's advantages. Thus, option D is overly restrictive. The RPM format doesn't impose any licensing requirements, contrary to option E. For more information, see Chapter 2.

**19.** D. The mv utility can be used to rename files as well as move them from one location to another, so option D is correct. The dd utility (option A) is used to copy files to backups, rm (option B) is used to remove (delete) files, cp (option C) copies files, and ln (option E) creates links. For more information, see Chapter 4.

**20.** B. Appending an ampersand (&) to a command causes that command to execute in the background. The program so launched still consumes CPU time, but it won't monopolize the shell you used to launch it. Thus, option B is correct. The start (option A) and background (option D) commands are fictitious. Although bg (option C) does place a job into the background, it doesn't launch a program that way; it places a process that has already been suspended (by pressing Ctrl+Z) into the background. The nice utility (option E) launches a program with modified priority, but a program so launched still monopolizes its shell unless you take additional steps. For more information, see Chapter 2.

**21.** A, B. The -Uvh parameter (option A) issues an upgrade command (which installs the program whether or not an earlier version is installed) and creates a series of hash marks to display the command's progress. The -i parameter (option B) installs the program if it's not already installed but causes no progress display. Option C uses a package name, not a complete filename, and so it will fail to install the package file. The -e option (option D) removes a package. Option E's -Vp option verifies the package file but doesn't install it. For more information, see Chapter 2.

**22.** B. Option B, fsck, is Linux's filesystem check utility. It's similar in purpose to the DOS and Windows CHKDSK and ScanDisk utilities (similar to options C and D), but these DOS and Windows utilities don't work on Linux filesystems like ext2fs or ReiserFS. Option A, mkfs, creates new filesystems; it doesn't diagnose or fix filesystem problems. Option E, fdisk, is a tool for creating or modifying disk partitions; it doesn't manage the filesystems they contain. For more information, see Chapter 3.

**23.** A. A freshly installed MySQL database is unlikely to have a ready-made database of animals, so your first task is to create that database with the CREATE DATABASE command, as shown in option A. (You could call the

database something other than animals, of course.) The USE command in option B will be useful only once the database has been created. Once the database is created, you can use CREATE TABLE, as in option C, to create a table; however, you'll need an existing database first, and this command

also requires information about the type of data to be stored, which option C doesn't provide. Option D's INSERT INTO command stores data into a table once it's been created, so it's far from the first command you'll use. It also requires additional specification of the data to be stored, so it's incomplete. Option E's UPDATE command modifies existing entries, so you'll use this command only after you've created the database and added at least one animal to it. (Option E is also an incomplete command even then.) For more information, see Chapter 9.

**24.** B, D. The correct answers, man and info (options B and D), are two common Linux help packages. Although ? (option C) is a common help command within certain interactive programs, it isn't a help command in bash or other common Linux shells. There is no common command called manual (option A), nor is hint (option E) a valid bash command or common program name. For more information, see Chapter 1.

**25.** A, C. Unix systems traditionally store time in UTC (aka Greenwich mean time), and Linux may do so as well. Thus, option A is correct. Most other *x*86 PC OSs traditionally store time as the local time, however, so Linux also supports this option and option C is also correct. Internet Time (option B) is an alternative to the 24-hour clock in which the day is broken into 1,000 “beats.” Standard PC BIOSs don't support this time format. Likewise, a 12-hour clock isn't terribly useful to computers because it doesn't differentiate a.m. from p.m., making option D incorrect. Although the length of the Martian day is similar to that of Earth (24 hours and 37 minutes), those wanting to colonize Mars will have to wait for PC clocks to support setting time for the Red Planet; option E is incorrect. For more information, see Chapter 7.

**26.** D. Typing **lsmod** (option D) produces a list of the modules that are currently loaded. The insmod (option A) and modprobe (option C) programs both load modules—either a single module or a single module and all those on which it depends, respectively. The depmod command (option B) generates the modules.dep file that contains module dependency information. The modinfo command (option E) displays information, such as its version number and author, on a single module. For more

information, see Chapter 3.

**27.** B, E. The chgrp and chown commands can both change the group ownership of a file. The chgrp command takes a group name and a filename as parameters, as in option B. The chown command normally changes a file's owner; but if you provide a group name preceded by a dot (.) or a colon (**:**), as in option E, it changes the group of a file. The chown command as used in option A, will change the primary ownership of the file to the music user, if such a user exists on the system; it won't change the group ownership. There is no standard chgroup command, as in option C. Option D will change the permissions to 0600 (-rw-------), which will be a step backward with respect to the goal stated. For more information, see Chapter 4.

**28.** E. Hard links to directories are not permitted by most filesystems, so you'll probably have to create a symbolic link, as noted in option E. Links don't rely on a filesystem journal, so option A is incorrect. Contrary to option B, anybody may create a link, not just the original's owner. Option C describes a restriction of hard links, but because this link will probably have to be a symbolic link, this restriction is unimportant and option C is incorrect. Option D describes a more severe restriction than option B, but it's incorrect for the same reasons. For more information, see Chapter 4.

**29.** B, E. The colon (:) starts ex mode, from which you can enter commands. In ex mode, r includes a file in an existing one, w writes a file, e loads an entirely new file, and q quits the program. Thus the desired combination is :wq (option B). As a special case, ZZ does the same thing, so option E is also correct. For more information, see Chapter 5.

**30.** C. The ∼/.forward file is a user email forwarding file. The vertical bar character (|) at the start of such a file is a code to send the email through the specified program file, so option C is correct. To do as option A describes, the file would need to read junkme or junkme@*hostname*, where *hostname* is the computer's hostname. To do as option B describes, the leading vertical bar would have to be omitted. It's conceivable that the ∼/junkme script does as option D describes, but there's no way of knowing this for certain. To do as option E describes, the file would have to read *user*@junkme, where *user* is the username. For more information, see Chapter 9.

**Part I**

**Exam LX0-103**

**Chapter 1**

**Exploring Linux Command-Line Tools**

**THE FOLLOWING EXAM OBJECTIVES ARE COVERED IN THIS CHAPTER:**

**103.1 Work on the command line**

**103.2 Process text streams using filters**

**103.4 Use streams, pipes, and redirects**

**103.7 Search text files using regular expressions**

Linux borrows heavily from Unix, and Unix began as a text-based operating system (OS). Unix and Linux retain much of this heritage, which means to understand how to use and, especially administer Linux, you must understand at least the basics of its command-line tools. Using command-line tools requires the use of a shell. A *shell* is a program that accepts and interprets text-mode commands and provides an interface to the system.

This chapter begins with basic shell information, including the various shell programs available and the procedures for using them. From there, this chapter covers streams, pipes, and redirection, which you can use to move input and output between programs or between files and programs. These techniques are frequently combined with text processing using *filters*—commands you can use to manipulate text without the help of a conventional text editor. Sometimes you must manipulate text in an abstract way, using codes to represent several different types of text. This chapter, therefore, covers this topic as well.

**Understanding Command-Line Basics**

Before you do anything else with Linux, you should understand how to use a Linux shell. The shell allows you to enter commands as needed. Which commands can be entered depends on which shell program is running. Several of the available shell programs are briefly described.

In using shell commands, you should also understand shell *environment variables*, which are placeholders for data that may be useful to many programs. Finally, it is helpful to know how to get help with the shell commands you're trying to use.

**Exploring Your Linux Shell Options**

The shell to be used for entering commands is configured for each individual user, and Linux provides a range of available shells. A complete shell list would be quite long, but the following shells are among the more common choices:

bash The GNU Bourne Again Shell (bash) is based on the earlier Bourne shell for Unix but extends it in several ways. In Linux, bash is the most common default shell for user accounts, and it's the one emphasized in this book and on the exam.

ShThe Bourne shell upon which bash is based goes by the name sh. It's not often used in Linux and the sh command is often a pointer to the bash shell or other shells.

tcsh This shell is based on the earlier C shell (csh). It's a fairly popular shell in some circles, but no major Linux distributions make it the default shell. Although it's similar to bash in many respects, some operational details differ. For instance, you don't assign environment variables the same way in tcsh as in bash.

csh The original C shell isn't used much on Linux, but if a user is familiar with csh, tcsh makes a good substitute.

ksh The Korn shell (ksh) was designed to take the best features of the Bourne shell and the C shell and extend them. It has a small but dedicated following among Linux users.

zsh The Z shell (zsh) takes shell evolution further than the Korn shell, incorporating features from earlier shells and adding still more.

In addition to these shells, dozens more obscure ones are available. In Linux, most users run bash because it is the most popular shell. Some other OSs use csh or tcsh as the default, so if your users have backgrounds on non-Linux Unix-like OSs, they may be more familiar with these other shells. You can change a user's default shell by editing their account, as described in Chapter 7, “Administering the System.”

Be aware that there are two types of default shells. The *default interactive shell* is the shell program a user uses to enter commands, run programs from the command line, run shell scripts, and so on. The other default shell type is a default *system* shell. The *default system shell* is used by the Linux system to run system shell scripts, typically at startup.

The file *bin*sh is a pointer to the system's default system shell—normally *bin*bash for Linux. However, be aware that, on some distributions, the *bin*sh points to a different shell. For example, on Ubuntu, *bin*sh points to the dash shell, *bin*dash.

**Using a Shell**

Linux shell use is fairly straightforward for anybody who's used a text-mode OS before: You type a command, possibly including options to it, and the computer executes the command. For the most part, Linux commands are external—that is, they're programs that are separate from the shell.

A few commands are internal to the shell, though, and knowing the distinction can be important. You should also know some of the tricks that can make using the command shell easier—how to have the computer complete a long command or filename, retrieve a command you've recently run, or edit a command you've recently used (or haven't yet fully entered).

**Starting a Shell**

If you log into Linux using a text-mode login screen, you have logged into a virtual console terminal and, most likely, you'll be dropped directly into your default shell. The shell program is what presents the prompt and accepts subsequent commands.

If you log into Linux using a graphical user interface (GUI) login screen, you'll have to start a terminal emulator manually in order to reach your default shell. Some GUIs provide a menu option, such as xterm or terminal, to start a terminal emulator program. These programs enable you to run text-mode

programs within Linux, and by default they come up running your shell. If you can't find such a menu option, look for a menu option that enables you to run an arbitrary command. Select it, and type **xterm** or **konsole** as the command name. This will launch a terminal emulator program that will run a shell.

Once you start a terminal or log into a virtual console terminal, the shell will provide you with a prompt for entering commands. Remember that the shell is a program providing you with an interface to the Linux system.

A good first command to try, uname, will show what operating system is being run:

$ **uname**

Linux

$

That's not too interesting. You can find out additional information by tacking on the -a option to the command. Be sure to include the necessary space between the command and the option:

$ **uname -a**

Linux server01.class.com 2.6.32-431.5.1.el6.x86\_64 #1 SMP Wed Feb 12

00:41:43 UTC 2014 x86\_64 x86\_64 x86\_64 GNU/Linux

$

The uname -a command provides a lot more information, including the current Linux kernel being used (2.6.32) as well as the system's hostname (server01.class.com). The uname command is an external command. The shell also provides internal commands. It's important to know the difference between the two command types, as explained in the next section.

**Using Internal and External Commands**

Internal commands are, as you might expect, built into the shell program. Thus they are also called *built-in commands*. Most shells offer a similar set of internal commands, but shell-to-shell differences do exist. Internal commands that you're likely to use enable you to perform some common tasks:

**Change the Working Directory**

Whenever you're running a shell, you're working in a specific directory. The cd command changes the current working directory. For instance, typing **cd *home*sally** changes the current working directory to the

*home*sally directory.

You can use shortcut characters with the cd command as well. The tilde (∼) character is a useful shortcut; it stands for your home directory. Thus typing **cd ∼** will have the same effect as typing **cd *home*sally** if your home directory is *home*sally.

**Display the Working Directory**

The pwd command displays (“prints” to the screen) the current working directory. This command is helpful, especially after you have changed your working directory, to ensure you ended up in the right place.

**Display a Line of Text**

The echo command displays the text you enter. For instance, typing **echo Hello** causes the system to display the string Hello. This may seem pointless, but it's useful in scripts (described in Chapter 9, “Writing Scripts, Configuring Email, and Using Databases”), and it can also be a good way to review the contents of environment variables (described later in this chapter, in the section “Using Environment Variables”).

**Time an Operation**

The time command times how long subsequent commands take to execute. For instance, typing **time pwd** tells you how long the system took to execute the pwd command. The time is displayed after the full command terminates. Three times are displayed: total execution time (aka real time), user CPU time, and system CPU time. The final two values tell you about CPU time consumed, which is likely to be much less than the total execution time.

**Set Options**

In its most basic form, the set command displays a wide variety of options relating to bash shell operation. These options are formatted much like environment variables, but they aren't the same things. You can pass various options to set to have it affect a wide range of shell operations.

**Terminate the Shell**

The exit and logout commands both terminate the shell. The exit command terminates any shell, but the logout command terminates only *login shells*. Login shells are shell programs that are launched automatically when you initiate a text-mode login as opposed to those that run in xterm

windows or other terminal emulators.



The preceding list isn't complete. Later sections of this chapter and later chapters describe some additional internal commands. Consult your shell's documentation for a complete list of its internal commands.

You can quickly determine if a command is a builtin command by using the type command. Just enter the command type before the name of the command you wish to check:

$ **type pwd**

pwd is a shell builtin

$

$ **type cd**

cd is a shell builtin

$

$ **type bash**

bash is *bin*bash

$

Some of these internal commands are duplicated by external commands that do the same thing. But those external commands aren't always installed on all systems. You can see if there are internal commands with installed duplicate external commands by using the -a option on the type command:

$ **type -a cd**

cd is a shell builtin

$

$ **type -a pwd**

pwd is a shell builtin

pwd is *bin*pwd

$

You can see that on this system, there is no external cd command installed. However, it does have an external pwd command installed.

Keep in mind that even when external commands are installed, the internal command takes precedence. To access the external command, you must provide the complete external command path, as in typing **/usr*bin*time** rather than **time**.



**Confusion over Internal and External Commands**

When duplicate internal and external commands exist, they sometimes produce subtly different results or accept different options. These differences may occasionally cause problems if you are unaware of them. For example, the time built-in command returns slightly different results than the *usr*bin/time external command: $ **time pwd** *home*Christine real 0m0.002s user 0m0.002s sys 0m0.001s $ $ ***usr*bin/time pwd** *home*Christine 0.00user 0.00system 0:00.04elapsed 24%CPU (0avgtext+0avgdata 2336maxresident)k 56inputs+0outputs (1major+173minor)pagefaults 0swaps $

As you can see, bash's internal time shows the time to execute the pwd command in a very nice format, while the external time command *usr*bin/time is not only a little sloppy in appearance, it also provides additional details. Be mindful of the potential behavior differences between internal and external commands.

When you type a command that's not recognized by the shell as one of its internal commands, the shell checks its path to find a program by that name to execute it. The *path* is a list of directories in which commands can be found. It's defined by the $PATH environment variable, as described shortly in “Using Environment Variables.” A typical user account has about half a dozen or so directories in its path. You can add and remove directories to the shell's path by changing the $PATH environment variable in a shell configuration file, as described in “Exploring Shell Configuration” later in this chapter.

You can run programs that aren't on the path by providing a complete path name on the command line. For instance, typing **./myprog** runs the myprog program in the current directory. Typing ***home*arthur/thisprog** runs the thisprog program in the *home*arthur directory.



The root account should normally have a shorter path than ordinary user accounts. Typically, you'll omit directories that store GUI and other user oriented programs from root's path in order to discourage use of the root

account for routine operations. This minimizes the risk of security breaches related to buggy or compromised binaries being run by root. Most important, root's path should never include the current directory (**.**/). Placing this directory in root's path makes it possible for a local troublemaker to trick root into running replacements for common programs. Omitting the current directory from ordinary user paths is also generally a good idea. If this directory must be part of the ordinary user path, it should appear at the end of the path so that the standard programs take precedence over any replacement programs in the current directory.

Whether you need to enter the path or not for a command, the program file must be marked as executable. This is done via the execute bit that's stored with the file. Standard programs are marked as executable when they're installed, but if you need to adjust a program's executable status, you can do so with the chmod command, as described in Chapter 4, “Managing Files.”

**Performing Some Shell Command Tricks**

Many users find typing commands to be tedious and error-prone. This is particularly true of slow or sloppy typists. For this reason, Linux shells include various tools that can help speed up operations. The first of these is *command completion*: Type part of a command or a filename (as an option to the command), and then press the Tab key. The shell tries to fill in the rest of the command or the filename. If just one command or filename matches the characters you've typed so far, the shell fills the rest of the command (or filename) for you and adds a space after it.

If the characters you've typed don't uniquely identify a command (or filename), the shell fills in what it can and then stops. Depending on the shell and its configuration, it may beep. If you press the Tab key again, the system responds by displaying the possible completions. You can then type another character or two and, if you haven't completed the command (or filename), press the Tab key again to have the process repeat.

again to have the process repeat.

The most fundamental Linux commands have fairly short names—mv, ls, set, and so on. However, some other commands are much longer, such as traceroute or service --status-all. Filenames can also be quite lengthy— up to 255 characters on many filesystems. Thus command completion can save a lot of time when you're typing. It can also help you avoid typos.



The most popular Linux shells, including bash and tcsh, support command and filename completion. Some older shells, though, don't support this helpful feature.

Another useful shell shortcut is *history*. The shell history keeps a record of every command you type. If you've typed a long command recently and want to use it again or use a minor variant of it, you can pull the command out of the history.

There are several rather easy methods to retrieve commands. It comes down to determining the method you like best:

**Retrieve a Command**

The simplest way to do this is to press the Up arrow key on your keyboard; this brings up the previous command. Pressing the Up arrow key repeatedly moves through multiple commands so you can find the one you want. If you overshoot, press the Down arrow key to move down the history. The Ctrl+P and Ctrl+N keystrokes double for the Up and Down arrow keys, respectively.

**Search for a Command**

Press Ctrl+R to begin a backward (reverse) search, and begin typing characters that should be unique to the command you want to find. The characters you type need not be the ones that begin the command; they can exist anywhere in the command. You can either keep typing until you find the correct command or, after you've typed a few characters, press Ctrl+R repeatedly until you find the one you want.

The Ctrl+S keystroke is used to search forward in the command history. You can press the Ctrl+S keystroke *while* using the backward search. This reverses the history search from backward to forward. If you used a backward search and have passed by what you need, then this keystroke is useful.



If the Ctrl+S keystroke causes your terminal to hang, press Ctrl+Q to resume terminal operations. To keep your terminal from hanging when Ctrl+S is used, type **stty -ixon** at the command line.

In either event, if you can't find the command you want or if you change your mind and want to terminate the search, press Ctrl+G to do so.

Frequently, after finding a command in the history, you want to edit it. The bash shell, like many shells, provides editing features modeled after those of the Emacs editor:

**Move within the Line**

Press Ctrl+A or Ctrl+E to move the cursor to the start or end of the line, respectively. The Left and Right arrow keys move within the line a character at a time. Ctrl+B and Ctrl+F do the same, moving backward and forward within a line. Pressing Ctrl plus the Left or Right arrow key moves backward or forward a word at a time, as does pressing Esc and then B or F.

**Delete Text**

Pressing Ctrl+D or the Delete key deletes the character under the cursor. Pressing the Backspace key deletes the character to the left of the cursor. Pressing Ctrl+K deletes all text from the cursor to the end of the line. Pressing Ctrl+X and then Backspace deletes all of the text from the cursor to the beginning of the line.

**Transpose Text**

Pressing Ctrl+T transposes the character before the cursor with the character under the cursor. Pressing Esc and then T transposes the two words immediately before (or under) the cursor.

**Change Case**

Pressing Esc and then U converts text from the cursor to the end of the word to uppercase. Pressing Esc and then L converts text from the cursor to the end of the word to lowercase. Pressing Esc and then C converts the letter under the cursor (or the first letter of the next word) to uppercase,

letter under the cursor (or the first letter of the next word) to uppercase, leaving the rest of the word unaffected.

**Invoke an Editor**

You can launch a full-fledged editor to edit a command by pressing Ctrl+X followed by Ctrl+E. The bash shell attempts to launch the editor defined by the $FCEDIT or $EDITOR environment variable, or it launches Emacs as a last resort.

These editing commands are just the most useful ones supported by bash. In practice, you're likely to make heavy use of command and filename completion, the command history, and perhaps a few editing features.



If you prefer the vi editor to Emacs, you can use a vi-like mode in bash by typing **set -o vi**. (vi is described in Chapter 5, “Booting Linux and Editing Files.”)

The history command provides an interface to view and manage the history. Typing **history** alone displays all of the commands in the history (typically the latest 500 commands).

To retrieve the last command in your shell history, type **!!** and press Enter. This will not only show you the command you recalled but execute it as well:

$ **!!**

type -a pwd

pwd is a shell builtin

pwd is *bin*pwd

$

You can execute a command by number via typing an exclamation mark followed by its number, as in **!210** to execute command 210. Typing **history - c** clears the history, which can be handy if you've recently typed commands you'd rather not have discovered by others, such as commands that include passwords.

The bash history is stored in the .bash\_history file in your home directory. This is an ordinary plain-text file, so you can view it with a text editor or a command such as less (described later, in “Paging through Files with less”).



Because your bash history is stored in a file, it can be examined by anybody who can read that file. Some commands enable you to type passwords or other sensitive data on the same line as the commands themselves, which can therefore be risky. The ∼/.bash\_history file does *not* record what you type in response to other programs' prompts, just what you type at the bash prompt itself. Thus, if you have a choice, you should let commands that require passwords (or other sensitive data) prompt you to enter this data rather than enter such information as options to the command at the bash prompt.

In Exercise 1.1, you'll experiment with your shell's completion and command editing tools.

**Exercise 1.1**

**Editing Commands**

To experiment with your shell's completion and command-editing tools, follow these steps:

**1.** Log in as an ordinary user.

**2.** Create a temporary directory by typing **mkdir test**. (Directory and file manipulation commands are described in more detail in Chapter 4.)

**3.** Change into the test directory by typing **cd test**.

**4.** Create a few temporary files by typing **touch one two three**. This command creates three empty files named one, two, and three.

**5.** Type **ls -l t** and, without pressing the Enter key, press the Tab key. The system may beep at you or display two three. If it doesn't display two three, press the Tab key again and it should do so. This reveals that either two or three is a valid completion to your command, because these are the two files in the test directory whose filenames begin with the letter t.