DAMIAN TOMMASINO

Cert Guide

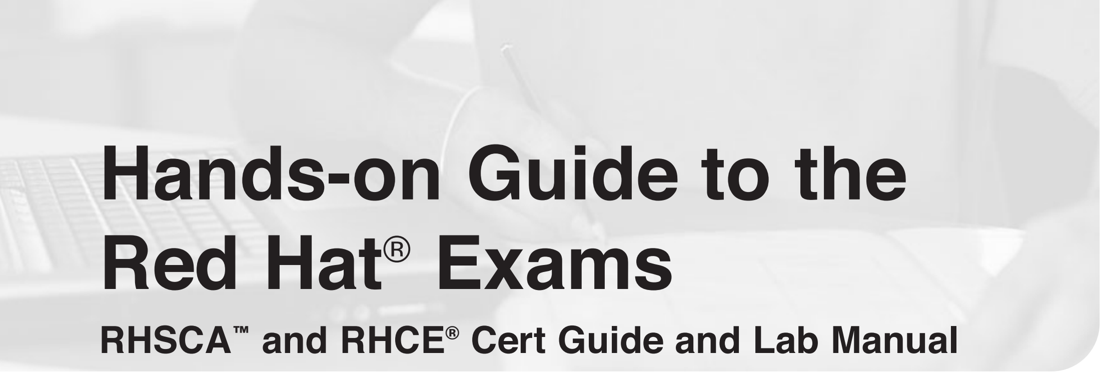
Learn, prepare, and practice for exam success

- Master every topic on Red Hat's new RHCSA™ and RHCE® exams.
- Assess your knowledge and focus your learning.
- Get the practical workplace knowledge you need!

Hands-on Guide to the Red Hat® Exams:

RHCSA™ and RHCE® Cert Guide and Lab Manual

PEARSON



Hands-on Guide to the Red Hat® Exams

RHSCA™ and RHCE® Cert Guide and Lab Manual

Damian Tommasino

Pearson
800 East 96th Street
Indianapolis, Indiana 46240 USA

Hands-on Guide to the Red Hat® Exams

Copyright © 2011 by Pearson Education, Inc.

ISBN-13: 978-0-321-76795-0
ISBN-10: 0-321-76795-0

Library of Congress Cataloging-in-Publication data is on file.
Printed in the United States of America

First Printing: May 2011

Trademarks

Warning and Disclaimer

Bulk Sales

Editor-in-Chief
Mark Taub

Executive Editor
Debra Williams Cauley

Senior Development Editor
Chris Zahn

Managing Editor
Sandra Schroeder

Project Editor
Mandie Frank

Copy Editor
Chuck Hutchinson

Indexer
Tim Wright

Proofreader
Megan Wade

Technical Editor
Robert P.J. Day

Editorial Assistant
Kim Boedigheimer

Book Designer
Gary Adair

Composition
Mark Shirar

Contents at a Glance

Table of Contents

About the Author

Damian Tommasino is currently a Linux system administrator at TradeCard and CEO of Modular Learning, Inc., an online IT training company. His current certifications include RHCE, RHCSA, MCSA, CCNA, CCENT, MCP, Security+, Network+, and A+. He has a popular blog called Security Nut (http://secnut.blogspot.com) that covers Red Hat, Linux, security, and more. Damian also spends time over at techexams.net helping out in the forums and conversing with friends.

Acknowledgments

I'd like to thank both Chris Zahn and Debra Williams Cauley at Pearson for all their help in making this book a reality. This book would not have been possible without them and the rest of the team at Pearson. It has been wonderful to work with both of them.

We Want to Hear from You!

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

As an Editor in Chief for Pearson IT Certification, I welcome your comments. You can email or write me directly to let me know what you did or didn't like about this book—as well as what we can do to make our books better.

Please note that I cannot help you with technical problems related to the topic of this book. We do have a User Services group, however, where I will forward specific technical questions related to the book.

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

Email: feedback@quepublishing.com

Mail: Mark Taub
Editor in Chief
Pearson IT Certification
800 East 96th Street
Indianapolis, IN 46240 USA

Reader Services

Visit our website and register this book at www.informit.com/title/9780321767950 for convenient access to any updates, downloads, or errata that might be available for this book.

This page intentionally left blank

Preface

This book was written as a lab guide to help individuals pass the RHCSA (EX200) and RHCE (EX300) exams. It is meant for those with different amounts of experience, from novice to expert, and is structured to make it easy for any reader to find what he is looking for. The book contains 22 chapters and two full-length lab exams.

Book Features

Each chapter includes the following elements to aid your learning:

- **Opening topics list**—This list defines the topics to be covered in the chapter; it also lists the corresponding Red Hat objectives.
- **Review Questions**—Review questions help reinforce what you learned and help you identify what you may need to review.
- **Answers to Review Questions**—Answers are provided for each of the review questions.
- **Labs**—Chapters conclude with several lab-based exercises that provide hands-on training and also help you to see what questions on the actual exam might be like.

The labs also include scripts that can help you with troubleshooting. The scripts use the following syntax:

- v_script_name   Used to verify a service or configuration
- t_script_name   Used to cause trouble on your system

I have also included two full-length labs at the end of the book intended to give you an experience like that of the real exam as well as examples of what the real exam might cover.

I have also produced an additional set of scripts that you can download that will purposely cause trouble on your system. You can download them from

- http://sourceforge.net/projects/rhcelabscripts/

Exam Registration and Costs

To register for the Red Hat exams, you must visit Red Hat's site at http://www.redhat.com/training and enroll online. The price for the new RHCSA exam is $399, and it is 2 hours in length. With the addition of the RHCSA certification, the price of the RHCE exam has been reduced to $399 (down from $799).

The RHCE exam is also 2.5 hours in length. Each exam is performance based, meaning it is given in the form of labs. With the addition of the RHCSA certification, you are now required to obtain the RHCSA before you can become RHCE certified. You can still take the RHCE exam; however, you will not receive the certification until you have completed and passed both exams.

LPIC, RHCE, and Other Things You Should Know

The Red Hat exams are a big undertaking, particularly if you have never taken a performance-based exam before. There is the unknown element of what to expect on the exam plus the amount of material you need to be familiar with. Before sitting for either of the Red Hat exams, you might want to consider completing the LPIC-1 exam series. Why? The Red Hat exams test experience and skill, not just your ability to memorize content within a book. There is also a certain skill set that you need before you take the Red Hat exams. You are expected to know basic Linux commands, to be able to navigate around a system, and to be able to perform basic file operations. Being able to effectively use some form of text editor is a good thing, too.

The LPIC-1 certification is broken down into two exams: LPIC 101 and LPIC 102. The material covered in both of these exams is equivalent to the knowledge a junior system administrator should have, and it gives you a solid foundation for taking on the Red Hat exam material. Although many topics overlap between the LPIC-1 material and the Red Hat exams, this will only help to reinforce your understanding of particular topics. You should look through the exam objectives of the LPIC-1 exams to gain a better understanding of some of the prerequisite skills required. The objectives for the LPIC-1 exams are very detailed, so they will help you identify any weak areas you might have:

- LPIC-101

  http://www.lpi.org/eng/certification/the_lpic_program/lpic_1/exam_101_detailed_objectives
- LPIC-102

  http://www.lpi.org/eng/certification/the_lpic_program/lpic_1/exam_102_detailed_objectives

If you already have a solid set of Linux skills, you should have no problem starting out here. If you don't, you can still proceed with this book but will need to put in some extra effort in areas you don't fully understand. One question I see frequently is, "Should I take the LPIC exams if I'm an RHCSA/RHCE?" My answer is always yes! The reason behind this is that the Red Hat exams are vendor specific, whereas the LPIC-1 exams are vendor neutral. They focus more on implementing services and working with Linux from an unbiased perspective. Holding both certifications adds diversity to your resume, and the exams shouldn't be hard to pass with the amount of overlap in the material between the Red Hat exams and the LPIC-1 exams.

You should know the following topics (prerequisites) before you start studying for the Red Hat exams. This is by no means a complete list!

- How to use a text editor (vim, emacs, or nano)
- File system hierarchy structure
- Different types of media (/dev/sda versus /dev/hda)
- File operations:

■ pwd	~	find	w
■ path	cat	locate	who
■ ls	more	cp	
■ echo	less	mv	
■ cd	tail	ln	
■ sort	head	wc	

- How to search with grep
- Command piping
- The basics of sed and awk
- Compression:
  - tar
  - gunzip
  - bzip2
- Networking basics:
  - ping
  - netstat
  - ifconfig
- IP addresses, subnets, and gateways
- How to use a command line and a GUI-based email client

If you lack the experience, the introduction to this book covers a majority of these prerequisite commands. Although it shouldn't count as a replacement for learning all these commands individually, the introduction can get you up to speed quickly if you have little to no current Linux experience.

Self-Study and Experience

One of the biggest debates I see among those studying for the Red Hat exams is, "Should I self-study or take a course?" I am a self-study person and have yet to find a halfway decent course for a price that wouldn't give a person a heart attack. The problem that most people seem to encounter with taking a course is the cost. Simply put, they are not cheap! The average price for a Red Hat training course is around $3,000, and such a course typically consists of four to five days of classroom training (which means footing the bill and taking time off work).

NOTE Red Hat offers an eLearning (or online version) of its training course for about half the price. I highly recommend that you DO NOT take this class because the learning experience is very different from that given in the classroom.

The benefits to taking a course, however, are that it is tailored specifically for the exams and the instructors can help you with questions. With the self-study option, you have to balance what you think important topics are (more likely to be tested on) versus less important topics (not likely to appear on the exam). This is really a strong point of the LPIC-1 exams: They list a "weight" for each topic, so you know how heavily it will count on the exam. If you spend the time researching the experience others have had on the Red Hat exams and read through the Red Hat Exam Prep Guide, you will start to get a feel for what topics are more likely to appear on the exam.

Experience plays another big factor in taking the Red Hat exams. After much research and talking to those who have taken the exams, I believe the amount of experience presented in Table P-1 would be required for each exam.

Table P-1 Experience Recommended for the Red Hat Exams

Exam	Years of Experience
LPIC-101/102	0–1 year
LPIC-201/202	2 years
RHCSA	2 years
RHCE	3 years

Although these are only my recommendations, you will probably find, with a little research on the Web, they are pretty accurate. As you probably know too, everyone is different and learns at different rates. The biggest difference between the two exams is that the Red Hat exams are all hands-on (performance based), whereas the

LPIC-1 exams are multiple choice. Unless you truly know what you are doing and have experience in the technologies listed in the Red Hat Exam Prep Guide, you will not pass the Red Hat exams. Don't worry, though, because a little experience (either at home or on the job) and some lab work will fix that. I hope that you will have both, which makes the learning process slightly easier and more rewarding.

Materials from Red Hat

No exam would be complete without a listing of what you should know. In Red Hat's case, the company has created a prep guide that lists the topics you need to know for the exams. With the release of Red Hat Enterprise Linux 6 and the addition of the RHCSA, the exam prep guide has become more specific about what you are required to know for the Red Hat exams. This book covers every topic you need to know for both exams. Before you begin studying, review the prep guide for each exam. If you don't have one printed out or saved already, you can get it here:

- Red Hat RHCSA Exam Prep Guide

  https://www.redhat.com/certification/rhcsa/objectives/

- Red Hat RHCE Exam Prep Guide

  https://www.redhat.com/certification/rhce/objectives/

I have also included a copy of each in the next two sections of this preface. If you have taken an earlier version of the RHCE, you may notice that the required objectives have become more specific about what you need to know. This is good because they leave you with less guessing to do. One of the great benefits of the Red Hat exams is that they don't list any specific technology that you must know. For example, if the exam requires that you block access to a particular service, you can choose to use TCP Wrappers, iptables, or the security of the service itself. This approach is good because, just as in the real world, there is always more than one way to do something. Another example might be the exams requiring you to set up outgoing mail using SMTP. You could use the Sendmail service or Postfix. As long as the system is allowed to send out mail, the exams don't care how you accomplish it. The only exception, of course, is unless the exams specifically ask you to use a particular service. These requirements will be useful as you study and practice for the exams in case you already have experience with a particular service.

EXAM TIP

It is worth noting that although you have some freedom on the exam to implement different technologies, Red Hat may ask you do something in a particular way. Going back to the example of blocking something on the system, you may use any method you like, unless Red Hat says that you need to specifically use iptables.

To aid you in setting up, configuring, and securing everything needed for the exams, Red Hat also provides documentation for its operating system. With the release of Red Hat Enterprise Linux 6, the documentation layout has also changed. The documentation guides are broken down into different sections instead of the two guides (installation guide and deployment guide) that were previously given. The following documentation is available from Red Hat:

- Installation Guide

  http://docs.redhat.com/docs/en-US/Red_Hat_Enterprise_Linux/6/html/Installation_Guide/index.html

- Managing Confined Services

  http://docs.redhat.com/docs/en-US/Red_Hat_Enterprise_Linux/6/html/Managing_Confined_Services/index.html

- Migration Planning Guide

  http://docs.redhat.com/docs/en-US/Red_Hat_Enterprise_Linux/6/html/Migration_Planning_Guide/index.html

- Security-Enhanced Linux

  http://docs.redhat.com/docs/en-US/Red_Hat_Enterprise_Linux/6/html/Security-Enhanced_Linux/index.html

- Security Guide

  http://docs.redhat.com/docs/en-US/Red_Hat_Enterprise_Linux/6/html/Security_Guide/index.html

- Storage Administration Guide

  http://docs.redhat.com/docs/en-US/Red_Hat_Enterprise_Linux/6/html/Storage_Administration_Guide/index.html

- Virtual Server Administration

  http://docs.redhat.com/docs/en-US/Red_Hat_Enterprise_Linux/6/html/Virtual_Server_Administration/index.html

- Virtualization Guide

  http://docs.redhat.com/docs/en-US/Red_Hat_Enterprise_Linux/6/html/Virtualization/index.html

You can find these guides available in HTML, EPUB, and PDF format. These guides are helpful tools when you're studying for the exams because they provide more command options than can be covered in any book. I recommend that you keep them close by as a reference.

RHCSA Exam Prep Guide

Understand and Use Essential Tools

- Access a shell prompt and issue commands with the correct syntax.
- Use input-output redirection (>, >>, |, 2>, etc.).
- Use grep and regular expressions to analyze text.
- Access remote systems using SSH and VNC.
- Login and switch users in multi-user runlevels.
- Archive, compress, unpack, and uncompress files using tar, star, gzip, and bzip2.
- Create and edit text files.
- Create, delete, copy, and move files and directories.
- Create hard and soft links.
- List, set, and change standard ugo/rwx permissions.
- Locate, read, and use system documentation including man, info, and files in /usr/share/doc.

Operate Running Systems

- Boot, reboot, and shut down a system normally.
- Boot systems into different runlevels manually.
- Use single-user mode to gain access to a system.
- Identify CPU and memory-intensive processes, adjust process priority with renice, and kill processes.
- Locate and interpret system log files.
- Access a virtual machine's console.
- Start and stop virtual machines.
- Start, stop, and check the status of network services.

Configure Local Storage

- List, create, delete, and set partition types for primary, extended, and logical partitions.
- Create and remove physical volumes, assign physical volumes to volumes groups, and create and delete logical volumes.
- Create and configure LUKS-encrypted partitions and logical volumes to prompt for password and be available at system boot.

- Configure systems to mount file systems at boot by using Universally Unique ID (UUID) or labels.
- Add new partitions, logical volumes, and swap to a system non-destructively.

Create and Configure File Systems

- Create; mount; unmount; and use ext2, ext3, and ext4 file systems.
- Mount, unmount, and use LUKS-encrypted file systems.
- Mount and unmount CIFS and NFS network file systems.
- Configure systems to mount ext4, LUKS-encrypted, and network file systems automatically.
- Extend existing unencrypted ext4 formatted logical volumes.
- Create and configure set-GID directories for collaboration.
- Create and manage access control lists (ACLs).
- Diagnose and correct file permission problems.

Deploy, Configure, and Maintain Systems

- Configure network and hostname resolution statically or dynamically.
- Schedule tasks using cron.
- Configure systems to boot into a specific runlevel automatically.
- Install Red Hat Enterprise Linux automatically using kickstart.
- Configure a physical machine to host virtual guests.
- Install Red Hat Enterprise Linux systems as virtual guests.
- Configure systems to launch virtual machines at boot.
- Configure network services to start automatically at boot.
- Configure a system to run a default configuration HTTP server.
- Configure a system to run a default configuration FTP server.
- Install and update software packages from the Red Hat Network, a remote repository, or from the local file system.
- Update the kernel package appropriately to ensure a bootable system.
- Modify the system bootloader.

Manage Users and Groups

- Create, delete, and modify local user accounts.
- Change passwords and adjust password aging for local user accounts.
- Create, delete, and modify local groups and group memberships.
- Configure a system to use an existing LDAP directory service for user and group information.

Manage Security

- Configure firewall settings using system-config-firewall or iptables.
- Set enforcing and permissive modes for SELinux.
- List and identify SELinux and file process context.
- Restore default file contexts.
- Use Boolean settings to modify system SELinux settings.
- Diagnose and address routine SELinux policy violations.

RHCE Exam Prep Guide

System Configuration and Management

- Route IP traffic and create static routes.
- Use iptables to implement packet filtering and configure network address translation (NAT).
- Use /proc/sys and sysctl to modify and set kernel run-time parameters.
- Configure a system to authenticate using Kerberos.
- Build a simple RPM that packages a single file.
- Configure a system as an iSCSI initiator that persistently mounts an iSCSI target.
- Produce and deliver reports on system utilization (processor, memory, disk, and network).
- Use shell scripting to automate system maintenance tasks.
- Configure a system to log to a remote system.
- Configure a system to accept logging from a remote system.

HTTP/HTTPS

- Install the packages needed to provide the service.
- Configure SELinux to support the service.

- Configure the service to start when the system is booted.
- Configure the service for basic operation.
- Configure host-based and user-based security for the service.
- Configure a virtual host.
- Configure private directories.
- Deploy a basic CGI application.
- Configure group-managed content.

DNS

- Install the packages needed to provide the service.
- Configure SELinux to support the service.
- Configure the service to start when the system is booted.
- Configure the service for basic operation.
- Configure host-based and user-based security for the service.
- Configure a caching-only name server.
- Configure a caching-only name server to forward DNS queries (forwarding server).

FTP

- Install the packages needed to provide the service.
- Configure SELinux to support the service.
- Configure the service to start when the system is booted.
- Configure the service for basic operation.
- Configure host-based and user-based security for the service.
- Configure anonymous-only downloads.

NFS

- Install the packages needed to provide the service.
- Configure SELinux to support the service.
- Configure the service to start when the system is booted.
- Configure the service for basic operation.
- Configure host-based and user-based security for the service.
- Provide network shares to specific clients.
- Provide shares suitable for group collaboration.

Samba

- Install the packages needed to provide the service.
- Configure SELinux to support the service.
- Configure the service to start when the system is booted.
- Configure the service for basic operation.
- Configure host-based and user-based security for the service.
- Provide network shares to specific clients.
- Provide shares suitable for group collaboration.

SMTP

- Install the packages needed to provide the service.
- Configure SELinux to support the service.
- Configure the service to start when the system is booted.
- Configure the service for basic operation.
- Configure host-based and user-based security for the service.
- Configure a mail transfer agent (MTA) to accept inbound email from other systems.
- Configure an MTA to forward (relay) email through a smart host.

SSH

- Install the packages needed to provide the service.
- Configure SELinux to support the service.
- Configure the service to start when the system is booted.
- Configure the service for basic operation.
- Configure host-based and user-based security for the service.
- Configure key-based authentication.
- Configure additional options described in documentation.

NTP

- Install the packages needed to provide the service.
- Configure SELinux to support the service.
- Configure the service to start when the system is booted.
- Configure the service for basic operation.
- Configure host-based and user-based security for the service.
- Synchronize time using other NTP peers.

Setting Up the Lab

Throughout this book, I show you how to use different systems to set up services, perform configurations, and implement security. In many forums I often see people asking how to set up labs or practice for the Red Hat exams. The lab used throughout this book is built completely on top of VirtualBox. VirtualBox is like VMware in that it allows you to virtualize systems. If you don't have VirtualBox, you should grab a copy because it is free to use and very helpful when practicing labs.

- VirtualBox

  http://www.virtualbox.org/wiki/Downloads

Because you will be using many different machines in the lab environment, Chapter 1 describes how to set up Red Hat Enterprise Linux (RHEL). You can install RHEL on your own or follow along in Chapter 1 to completely set up the lab.

Table P-2 presents a layout of the lab used here. Each ID is a different virtual machine.

Table P-2 Lab Layout

ID	Hostname	Red Hat Version	IP Address	Network
1	RHEL01	RHEL6	DHCP	Bridged
			172.168.1.1	Internal
2	RHEL02	RHEL6	172.168.1.2	Internal
3	Client01	RHEL5	172.168.1.10	Internal
4	Client02	RHEL6	172.168.1.20	Internal

As you can see, four machines are used. The first is a dual-homed server that also serves as the gateway for all the internal clients. A majority of the configuration work takes place on this server, and you use the second server (RHEL02) as a backup. The two client machines are to simulate users on the network. The reason I set up the network like this for you is that all testing and configuration are done in a controlled environment (which is a good habit to get into). If something ever happens on the internal network, it doesn't affect the rest of the external (home) network. Some other details for the lab setup include those shown in Table P-3.

Table P-3 Lab Layout

Host	Drive	Size	Layout
RHEL01	Disk 1	20GB	Default
	Disk 2	8GB	
	Disk 3	8GB	
	Disk 4	8GB	
RHEL02	Disk 1	10GB	Default
RHEL02	Disk 1	10GB	Default
Client01	Disk 1	10GB	Default

All virtual machines use 384MB of RAM for memory. I also disabled the sound device for each virtual system because I never use it, but that is entirely up to you.

NOTE All drives in VirtualBox are considered IDE and use the /dev/hdx format.

In the first chapter, you set up each virtual machine for the lab. If you have experience working with VirtualBox, you can set up your lab with the outlined requirements beforehand; otherwise, you can follow along in the first chapter.

No network is complete without documentation and a diagram to finally tie it all together. The network is represented in Figure P-1.

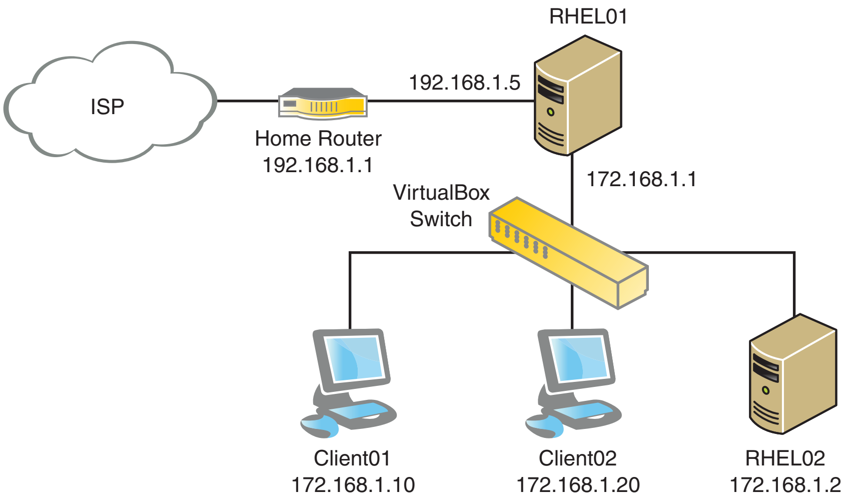


Figure P-1 The network diagram.

NOTE In case you're thinking you don't have the hardware to host this number of machines or you don't know how you'll virtualize an entire lab, think again. These four virtual machines each use 384MB of RAM (1.5GB total). The host machine that I use is a laptop so that my lab is portable, and it has a dual-core processor with 3GB of RAM. I have also tested this lab setup on a Pentium 4 with 4GB of RAM. Both host machines were able to run the full virtual lab with no problems or delays. If you have some trouble with performance, you can also drop the amount of memory on RHEL02, Client01, or Client02 to 256MB of RAM. The primary host (RHEL01) is the only machine that really needs the extra memory.

CAUTION Don't create the three 4GB drives for host RHEL01 just yet! One of the limitations in VirtualBox is that you can have only four devices attached to a system at one time. To get the operating system installed, you need to have a CD-ROM device attached, and if you create the four drives listed here, you will have no room left for the CD-ROM. After you complete Chapter 1, you can remove the CD-ROM device and create the three extra 4GB drives that you will need later.

Who Should Read This Book?

The Red Hat exams are some of the most challenging exams in the Linux arena. This book is meant to be used as a hands-on lab guide to readers with all types of backgrounds. Whether you are just starting out or are a seasoned system administrator, this book helps you learn or fine-tune your skills to take the Red Hat exams. Although those just starting out need to put in more effort to learn some of the skills discussed in this book, it is possible to gain the required skills for the exams. While this book teaches you the necessary skills, the key to passing the Red Hat exams is practice, practice, practice.

How This Book Is Organized

This book is laid out in a logical format that flows from cover to cover. Although you could jump around, each chapter builds on where the previous one left off, allowing you to build a system and understand how it works from the ground up. Although each chapter covers a different set of exam objectives, the first half of the book (Chapters 2 through 12) deals primarily with the RHCSA exam. The second half of the book (Chapters 13 through 21) covers the RHCE exam.

Chapter 1, "Installation," is an introductory chapter designed to help you install the Red Hat Enterprise Linux operating system and set up your virtual lab. The virtual lab that you set up will help you with the labs in each chapter, allowing you to build your hands-on skills for the real exams.

The first half of the book, Chapters 2 through 12, covers the following topics:

- **Chapter 2, "System Initialization"**—This chapter focuses on how to manage system services, system runlevels, and everything that occurs during the boot process. It also looks at how services work and are started and stopped.
- **Chapter 3, "Disks and Partitioning"**—This chapter addresses partitioning Red Hat systems. It discusses basic partitions, LVM, and RAID. Also covered are swap partitions and advanced use of LVM for in-depth storage management. This chapter prepares you to work with file systems in Chapter 4.
- **Chapter 4, "File Systems and Such"**—This chapter follows up where Chapter 3 left off. It describes file systems, how they work, and how to manage them. Also discussed are the new LUKS encryption options and file system security.
- **Chapter 5, "Networking"**—This chapter is all about networks. Nothing can happen unless you can communicate with other systems. This chapter describes how to set up and troubleshoot network connections and client-side DNS problems.
- **Chapter 6, "Package Management"**—This chapter examines how to install, search for, and remove software from Red Hat systems. It covers many different ways to work with packages, including building your own packages and package repositories.
- **Chapter 7, "User Administration"**—No system would be complete without users. This chapter covers user administration (creating, managing, and deleting). Also covered are switching between users and client-side authentication.
- **Chapter 8, "Network Installs"**—To make life easier, you can use automated installations. This chapter covers kickstart and how it can aid in the installation of Red Hat Enterprise Linux. Also covered is hands-free installation with DHCP and PXE boot clients.
- **Chapter 9, "System Logging, Monitoring, and Automation"**—This chapter dives into system logging and monitoring and how to interpret that data. It looks at different ways to find problems (or their answers). Also discussed is the automation of system monitoring.
- **Chapter 10, "The Kernel"**—This chapter discusses updating and tuning the kernel properly. Although the kernel is not a huge topic, it is important to address critical security issues with any system.
- **Chapter 11, "SELinux"**—This chapter covers one of the most complex topics in the book. It describes how to set up and work with SELinux without giving you a headache. Also covered is how to work with SELinux Boolean values to allow services to run properly.

- **Chapter 12, "System Security"**—This chapter talks all about system security, including TCP Wrappers, firewall rules, and security policies. Because firewall rules play a heavy role in all services, the second half of the book covers this topic in particular.

The second half of the book, Chapters 13 through 21, covers the following topics:

- **Chapter 13, "Remote Access"**—This chapter demonstrates how to remotely and securely manage your Red Hat systems. It covers SSH, the most popular remote management tool in Linux. Also covered is VNC for remote desktop management.
- **Chapter 14, "Web Services"**—This chapter discusses how to set up and manage Apache web servers. Because it is the most widely deployed web server in the world, this is a big topic in the Linux arena. This chapter also covers the Squid web proxy and how to use it in conjunction with Apache.
- **Chapter 15, "NFS"**—This chapter discusses network file systems. A great choice for centralized storage, NFS has many benefits over its SMB and FTP counterparts. Also covered in this chapter is connecting clients to NFS servers.
- **Chapter 16, "Samba"**—This chapter discusses Samba and how to set it up. As Samba progresses more and more, integration with Windows becomes easier for Linux systems. The chapter describes how to set up basic shares and printer services for Windows and Linux systems.
- **Chapter 17, "FTP"**—This chapter explains how to set up and use an FTP server. FTP is great for sharing files both securely and insecurely. The chapter describes the benefits of both, including how to troubleshoot FTP issues.
- **Chapter 18, "DNS"**—This chapter discusses how DNS works, server setup, and management of DNS servers. Although this is one of the most complex topics in the book, it is one of the easiest to work with after you understand it. This chapter also delves into different types of DNS servers.
- **Chapter 19, "Network Services"**—This chapter discusses setting up the core network services for your network. Topics include DHCP servers, NTP for time management, and more.
- **Chapter 20, "Email Services"**—This chapter explains how to properly set up different types of mail servers. Because email is one of the most critical business components, it is essential to understand how to work with this technology. The chapter also covers how to secure your mail servers so you don't get overrun by spammers.
- **Chapter 21, "Troubleshooting"**—This chapter discusses different troubleshooting steps for a variety of topics. Although this chapter doesn't cover all troubleshooting topics discussed throughout the book, it does cover the big topics that you should know for the exam.

The last chapter deals with Red Hat's newest addition, virtualization:

- **Chapter 22, "Virtualization with KVM"**—This chapter discusses how to use virtualization with Red Hat Enterprise Linux 6. It talks about installation, setup, and configuration of virtual machines. Also discussed is how to monitor your virtual machines when they are in use.

Also included are two full exams that simulate what the real exams are like. The lab activities will help you prepare by asking you to accomplish various tasks, which is very similar to the real exam. There is one practice exam for each of the Red Hat exams this book covers. If you can comfortably make it through the full exams in the allotted time, then you should be in good shape for the real exam! In addition to the 22 chapters and 2 full labs, this book provides end of chapter questions and tasks to help you prepare for the exam. There are also additional troubleshooting scripts available for download at http://sourceforge.net/projects/rhcelabscripts.

This introduction covers the following subjects:

- **File and Directory Management**—This section explains how to navigate, create, move, and explore files and directories on the system.
- **File Permission Basics**—This section explores file permissions and how the system uses them.
- **Using a Text Editor**—This section covers using a text editor effectively from the command line.
- **Regex**—This section covers regular expressions and how they are used for pattern matching.
- **I/O Redirection**—This section covers how to pipe commands and redirect output.
- **Compression and Archiving**—This section explains how to compress and archive files and directories.

The following RHCSA exam objectives are covered:

- Access a shell prompt and issue commands with the correct syntax
- Use input-output redirection (>, >>, |, 2>, and so on)
- Use `grep` and regular expressions to analyze text
- Archive, compress, unpack, and uncompress files using `tar, star, gzip,` and `bzip2`
- Create and edit text files
- Create, delete, copy, and move files and directories
- Create hard and soft links
- List, set, and change standard ugo/rwx permissions
- Locate, read, and use system documentation including man, info, and files in /usr/share/doc

Introduction

Everyone has to start somewhere, and Linux administrators and engineers are no exception. If you have purchased this book, I imagine that your goal is to pass the Red Hat exams (RHCSA and RHCE) while acquiring or improving your current Linux skills. This introduction covers user-level commands that you will be required to know before you embark on your journey of becoming a system administrator or engineer. These skills and commands are all essential for knowing how to work with Linux, not just Red Hat. Although the current *Red Hat Exam Prep Guide* doesn't list (and can't) all the commands covered in this introduction, everything covered here is required for you to get through the rest of this book. This is in no way a complete list of every user-level command, but it is everything you need to get started. Many of the topics here are also covered later in the book. If you already have a decent set of Linux skills, most of this introduction will probably be a review for you.

File and Directory Management

For you to be able to work with different parts of the system, you need to know how to get around the system! In this section, we look at the following basic commands:

`ls`	Displays the contents of a directory
`cp`	Copies files or directories from one location to another
`mv`	Moves or renames files and directories
`cd`	Changes the current location
`rm`	Deletes files or directories
`touch`	Creates empty files
`mkdir`	Creates a directory
`pwd`	Shows the present working directory
`file`	Displays the type of a file
`head`	Displays the beginning of a file
`tail`	Displays the end of a file

This might seem like a lot of commands to start with; however, they are all quite simple to use, which makes explaining them easy. First, let's start by displaying a directory on the system using the `ls` command.

Step 1. List the contents of the current directory:

```
# ls
Desktop  Documents  Downloads  Music  Pictures  Public  Templates
Videos
```

Although you can't tell because this book is black and white, these directories are displayed in blue, alerting the user to the fact that they are indeed directories. Although it's great to have all these directories, how do you know where you are on the system? To view your current location, you can use the `pwd` command.

Step 2. Show the current location:

```
# pwd
/home/user01
```

Presently, you are in user01's home directory, so the output of the `ls` command was all directories that belong to user01. Let's move out of user01's home directory into one of the subdirectories. Using the `cd` command, you can move between different directories.

Step 3. Move down one level into the Documents directory:

```
# cd Documents/
```

REAL-WORLD TIP
The trailing slash (/) is optional when you're using the `cd` command. It indicates that the name being specified is a directory.

Step 4. Now view the current location again:

```
# pwd
/home/user01/Documents
```

What if you want to move up one level to the directory you just came from? If you use `ls`, you don't see the user01 directory listed. You can, however, view two special directories.

Step 5. View all hidden directories with the `ls -a` command:

```
# ls -a
. ..
```

Notice what seems like just a bunch of dots? They actually stand for two special types of directories. The first—the single .—stands for the current directory. The second—double ..—is the directory above where you currently are located.

Step 6. To get back to the previous user01 directory, use the following:

```
# cd ..
```

Step 7. Verify with the `pwd` command:

```
# pwd
/home/user01
```

Now you should be able to navigate around the system.

Let's move on to creating files and directories. First, let's look at directory creation.

Syntax: `mkdir [option] DIRECTORY`

Options:

`-p`	Creates a parent directory as needed
`-v`	Provides verbose output

Step 1. Create a new directory called test:

```
# mkdir test
```

Step 2. Create a new set of directories within one another:

```
# mkdir -p another/quick/test
```

Because none of the directories you just chose exist, they are all created, including the subdirectories named quick and test.

Step 3. Verify the directory creation with the `ls` command:

```
# ls another
quick
```

```
# cd another

# ls
Test
```

Step 4. Return to the home directory:

```
# cd /home/user01
#pwd
/home/user01
```

As you will see throughout this book, there are a lot of quick tricks to navigating the system. Because the directories were all created successfully, let's move on to files. Using the `touch` command, you can create blank files.

Step 5. Create a file called test1:

```
# touch test1
```

Step 6. Verify its existence:

```
# ls
test1
```

Sometimes files need to be created before you can use them, which is why the `touch` command is useful. You might also want to use a blank file as a placeholder for something later. If you are ever unsure what type of file something is, you can use the `file` command to find out.

Step 1. Check the file type of test1:

```
# file test1
test1: empty
```

Step 2. Check the type of the password file on the system:

```
# file /etc/passwd
passwd: ASCII text
```

Along with being able to create and determine file types, you need to be able to read them as well. There are many times, however, when you don't need to read the whole file (think log files), but instead can just view a few entries from that file. Using the `tail` and `head` commands, you can view either the beginning or the end of a file.

Syntax: `head [options] FILE`

Options:

`-n`	Specifies the number of files to print
`-v`	Provides verbose output

Syntax: `tail [options] FILE`

Options:

`-n`	Specifies the number of files to print
`-f`	Continuously displays the end of file (useful for logs)
`-v`	Provides verbose output

Step 1. View the beginning of the messages log file:

```
# head /var/log/messages
Dec 5 03:13:06 RHEL01 dhclient: DHCPREQUEST on eth0 to
172.27.100.163 port 67
Dec 5 03:13:10 RHEL01 dhclient: DHCPREQUEST on eth0 to
255.255.255.255 port 67
Dec 5 03:13:20 RHEL01 dhclient: DHCPREQUEST on eth0 to
172.27.100.163 port 67
Dec 5 03:13:29 RHEL01 dhclient: DHCPREQUEST on eth0 to
255.255.255.255 port 67
Dec 5 03:13:30 RHEL01 dhclient: DHCPREQUEST on eth0 to
172.27.100.163 port 67
Dec 5 03:13:43 RHEL01 dhclient: DHCPREQUEST on eth0 to
255.255.255.255 port 67
Dec 5 03:13:44 RHEL01 dhclient: DHCPREQUEST on eth0 to
172.27.100.163 port 67
Dec 5 03:13:50 RHEL01 dhclient: DHCPREQUEST on eth0 to
255.255.255.255 port 67
```

Step 2. View the end of the messages log file:

```
# tail /var/log/messages
Dec 11 08:11:04 RHEL01 dhclient: DHCPDISCOVER on eth0 to
255.255.255.255 port 67 interval 13
Dec 11 08:11:04 RHEL01 dhclient: DHCPOFFER from 172.27.100.163
Dec 11 08:11:04 RHEL01 dhclient: DHCPREQUEST on eth0 to
255.255.255.255 port 67
Dec 11 08:11:04 RHEL01 dhclient: DHCPACK from 172.27.100.163
Dec 11 08:11:04 RHEL01 NET[26281]: /sbin/dhclient-script : updated
/etc/resolv.conf
Dec 11 08:11:04 RHEL01 dhclient: bound to 172.27.100.226 — renew-
al in 40864 seconds.
Dec 11 08:18:00 RHEL01 abrt[26389]: saved core dump of pid 26388
(/usr/libexec/fprintd) to /var/spool/abrt/ccpp-1292073480-
26388.new/coredump (757760 bytes)
Dec 11 08:18:00 RHEL01 abrtd: Directory 'ccpp-1292073480-26388'
creation detected
Dec 11 08:18:00 RHEL01 abrtd: Crash is in database already (dup of
/var/spool/abrt/ccpp-1291114420-26066)
```

You can see that being able to look at different sections of a file without actually opening it is really useful, particularly when it comes to looking at log files. Now that you know where one of the log files is, why don't you copy it to the /home/user01 directory? You can use the `cp` command for this.

Syntax: `cp [options] SOURCE DEST`

Options:

`-R`	Copies recursively
`-v`	Provides verbose output

Step 1. Copy the log file into the user01 home directory:

```
# cp /var/log/messages /home/user01
```

Step 2. You also could use the following:

```
# cp /var/log/messages .
```

Remember that the dot (.) represents the current location. After the messages log file is copied over, you should probably rename it for safekeeping. You can use the `mv` command to accomplish this in addition to moving the file to a new location.

Syntax: `mv [options] SOURCE DEST`

Options:

`-v`	Provides verbose output

Step 3. Rename the file by specifying the filename and the new name of the file:

```
# mv messages messages.bak
```

Step 4. With the file renamed, move it to the test directory for safekeeping:

```
# mv messages.bak test/
```

Because you specified a directory this time, the file was moved instead of renamed. You can also verify that the file was moved correctly.

Step 5. List the contents of the test directory:

```
# ls test
messages.bak
```

What if you moved this file by mistake? In that case, you would need to delete it to make room for a new one. You can use the `rm` command to remove files or directories.

Syntax: `rm [options] FILE`

Options:

`-R`	Deletes recursively
`-f`	Forces deletion
`-v`	Provides verbose output

Step 1. Delete the messages.bak file:

```
# cd test
# rm messages.bak
rm: remove regular file `messages.bak'? y
```

Notice that you are prompted to delete the file? By using the `-f` option, you can skip the confirmation. While you're deleting things, also remove the test directory.

Step 2. Delete the test directory:

```
# cd ..
# rm -Rf test/
```

Because this example uses the `-f` option, there is no confirmation and the directory is just deleted. Here's one thing you need to make a note of: You must use the `-R` option to delete directories. If you don't use the `-R` option, you get a warning message about the directory not being empty.

By now, you should be able to get around the system, create and remote files and directories, and view files and their types. It may seem as though we moved fast through this section, but these are really basic commands and anyone with any Linux experience should know most of them.

File Permission Basics

Just like every operating system, Linux comes with a set of permissions that it uses to protect files, directories, and devices on the system. These permissions can be manipulated to allow (or disallow) access to files and directories on different parts of the system. Here are some of the commands you can use to work with permissions:

`chmod`	Changes the permissions of files or directories
`chown`	Changes the owner and group of files or directories
`ls -l`	Displays file permissions and ownership of files or directories
`ll`	Same as `ls -l`
`umask`	Defines or displays the default permissions for creation of files or directories

Before starting to use commands, let's look at how permissions work first. Linux permissions are implemented through the properties of files and defined by three separate categories. They are broken down into the following:

User	Person who owns the file
Group	Group that owns the file
Other	All other users on the system

Permissions in Linux can be assigned one of two ways. You can use the mnemonic or a single digit to represent the permissions level.

Operation	Digit	Mnemonic	Description
Read	r	4	View file contents
Write	w	2	Write to or change
Execute	x	1	Run the file

Let's start by looking at file permissions before changing them. You can view the home directory from one of the users I have set up. To view a file's or directory's permissions, you can use two different commands.

Step 1. View file permissions for user01's home directory:

```
# ll /home/user01
total 32
drwxr-xr-x. 2 user01 user01 4096 Dec 11 07:43 Desktop
drwxr-xr-x. 2 user01 user01 4096 Dec 11 07:43 Documents
drwxr-xr-x. 2 user01 user01 4096 Dec 11 07:43 Downloads
-rw-rw-r--. 1 user01 user01    0 Dec 11 07:44 file1
-rw-rw-r--. 1 user01 user01    0 Dec 11 07:44 file2
drwxr-xr-x. 2 user01 user01 4096 Dec 11 07:43 Music
drwxr-xr-x. 2 user01 user01 4096 Dec 11 07:43 Pictures
drwxr-xr-x. 2 user01 user01 4096 Dec 11 07:43 Public
drwxr-xr-x. 2 user01 user01 4096 Dec 11 07:43 Templates
drwxr-xr-x. 2 user01 user01 4096 Dec 11 07:43 Videos
```

Step 2. You could also use the following:

```
# ll -s /home/user01
total 32
drwxr-xr-x. 2 user01 user01 4096 Dec 11 07:43 Desktop
drwxr-xr-x. 2 user01 user01 4096 Dec 11 07:43 Documents
drwxr-xr-x. 2 user01 user01 4096 Dec 11 07:43 Downloads
-rw-rw-r--. 1 user01 user01    0 Dec 11 07:44 file1
-rw-rw-r--. 1 user01 user01    0 Dec 11 07:44 file2
drwxr-xr-x. 2 user01 user01 4096 Dec 11 07:43 Music
drwxr-xr-x. 2 user01 user01 4096 Dec 11 07:43 Pictures
drwxr-xr-x. 2 user01 user01 4096 Dec 11 07:43 Public
drwxr-xr-x. 2 user01 user01 4096 Dec 11 07:43 Templates
drwxr-xr-x. 2 user01 user01 4096 Dec 11 07:43 Videos
```

Here, you can see 10 different options that can be set on the file. The first field determines whether it is a directory. Desktop is a directory as denoted by the `d` in the first field. The next three fields each represent the owner, the group, and other. For example, file1 is not a directory, the owner of the file can read and write to this file, the group can read and write, and all others can only read the file. You can also see that file2 has the same permissions. The owner and group are listed after the permissions, and for all files in this directory, the owner is user01 and the group is user01. Users and groups are used to control who has access to files and directories.

Now that you know how to view the permissions of files and directories, let's look at how to modify them. Let's start with changing the owner and group of a file using the `chown` command. There is another user called user02 you can use.

Syntax: `chown [option] [user] [: [group]]`

Options:

`-R`	Acts recursively
`-v`	Provides verbose output

Step 3. Change the owner of file1 from user01 to user02:

```
# chown user02 file1
```

Step 4. Change the group of file2 from user01 to user02:

```
# chown :user02 file2
```

Step 5. Now check the permissions again:

```
# ll
-rw-rw-r--. 1 user02 user01    0 Dec 11 07:44 file1
-rw-rw-r--. 1 user01 user02    0 Dec 11 07:44 file2
```

You can see that the user for file1 and group for file2 were changed appropriately. Currently, both files are set up to allow only other users to read them. What if you want to allow all users on your system to make changes to this file? In that case, you can use the `chmod` command to change permissions.

Syntax: `chmod [options] FILE`

Options:

`-R`	Acts recursively
`-v`	Provides verbose output

Step 6. Change the permissions in the "other" section to allow write access to this file:

```
# chmod 666 file1
```

Step 7. View the permissions change:

```
# ll
-rw-rw-rw-. 1 user02 user01    0 Dec 11 07:44 file1
```

Now user02, group user01, and everyone else on the system have read and write access to this file. You might be wondering where I got 666 for the permissions? When you assign permissions with numerical values, you add up the values of the permissions for each section. Because I wanted to keep the user and group permissions the same, I needed to make sure that they each had read (4) and write (2) permissions (4 + 2 = 6). Because I want the "other" group to be the same, I just continued with the same numerical value, which is where the 666 permissions come from. Being able to manipulate permissions isn't a difficult task but may take some practice to understand at first.

The last command that we look at is `umask`. You may be wondering where the permissions came from when we first created file1 and file2? They were given 664 as their default permissions upon creation, but where did that number come from? When users are created, they are assigned a `umask` value that defines all permissions for files and directories that users create. Let's look at how that works.

When files are created, they are given the default permissions 666 or `rw-rw-rw-` and directories are given the default permissions 777 or `rwxrwxrwx`. The `umask` command takes the default permissions and modifies them according to its mask value (through subtraction). Here is an example of how they are calculated:

File's default permissions	666 = 110 110 110 (in binary)
Subtract the umask	002 = 000 000 010 (in binary)
Value you get afterward	664 = 110 110 100 (in binary)

Now you have the default permissions 664 as shown previously. This calculation is the same with directories:

Directory's default permissions	777 = 111 111 111 (in binary)
Subtract the umask	002 = 000 000 010 (in binary)
Value you get afterward	775 = 111 111 101 (in binary)

Now you have the default permissions 775 as shown previously.

Step 8. You can display the current value of the `umask` to find out what it is set to by calling the `umask` command:

```
# umask
0002
```

All files and directories have a leading 0, which is used for more advanced permissions. For now, just know that it is there but that you can leave it off when calculating directory permissions. You can also supply a value to `umask` to be able to change it, but this is not recommended.

If you don't understand the information covered here, that's all right because Chapter 7, "User Administration," deals with users, user creation, and advanced file permissions more closely. However, you should know how to set file permissions and understand how they are calculated before moving on.

Using a Text Editor

Being able to use a text editor is probably one of the most critical skills to have as a system administrator. You constantly need to edit config files, write scripts, or make changes to system files...all of which require you to use a text editor. The three most popular editors available today include

`vi` or `vim`	Text editor with great flexibility
`emacs`	Similar to `vi`, an advanced text editor with many features
`nano`	A basic text editor for quick editing

For the Red Hat exams, you need to know how to use the `vim` text editor. It is installed by default with Red Hat Enterprise Linux 6, although it isn't the only editor available. `Vim` is an enhanced version of `vi` with syntax highlighting for programming. You can install additional text editors if the packages for them are available during the exam. Personally, I use `nano` for most of my quick edits and `vim` for any config file building from scratch or programming.

Let's look at some of the options for `vim`.

Syntax: `vim [arguments] [file]`

Arguments:

`-R`	Opens a file in read-only mode
`-b`	Specifies binary mode
`+`	Starts at the end of the file
`+<num>`	Starts at line `<num>`

`Vim` functions in three different modes: command mode, insert mode, and last line mode. When you first start working with a file, you are in command mode. Here, you can issue commands that allow you to move around your file without actually inserting text into the file. When you're ready to insert text into the file, you can use `i` or `a` to move into insert mode, enabling you to insert text at the cursor location. To move out of insert mode back into command mode, just press the Esc key. The last mode you can enter is last line mode, which you enter by typing a colon (`:`). In this mode, you can issue additional commands to save, quit, and do even more.

There are some additional options you should know for command mode.

Commands for command mode:

`e`	Moves to the end of a word
`b`	Moves to the beginning of a word
`$`	Moves to the end of a line
`H`	Moves to the first line onscreen
`L`	Moves to the last line onscreen
`i`	Enters insert mode
`a`	Appends after the cursor
`o`	Opens a new line below and inserts
`O`	Opens a new line above and inserts
`R`	Enters insert mode but replaces characters instead of inserting
`dd`	Deletes the current line
`x`	Deletes text under the cursor
`yy`	Yanks (copies) the current line
`p`	Pastes the yanked line
`u`	Undoes the last action

Commands for last line mode:

`:n`	Jumps to line `n`
`:w`	Writes the file to disk
`:q`	Quits `vim`
`:q!`	Quits without saving changes

Using `vim` is rather confusing at first if you have not used it before. I suggest you create and work with some sample files before continuing so that you become more familiar with it. You can also practice with `nano` and `emacs` to see the difference in how the editors work. Although `nano` is quicker to pick up and easier to use, it is not as flexible or powerful as `vim` or `emacs`. There is a huge war between system administrators over which text editor is better—`vim` or `emacs`. I won't choose sides, but I will say you should try both and see which you are more comfortable with.

Regex

From time to time, you need to hunt down something specific on your system, whether it's a file itself, something within a file, or through a script. In Linux, you can use regular expressions (also known as *regex*) that enable you to find specific information. Regex uses special expressions in combination with any of the following:

Literal	Any character used in a search or matching expression
Metacharacter	One or more special characters with special meaning
Escape sequence	Use of metacharacters as a literal

Regex is one of those topics that either you pick up or it causes tons of frustration and headaches. Just like anything else though, the more you practice and look at examples, the more you should begin to pick up on how it works and how to use it more efficiently.

In this section, we look at only a single command called `grep`, which can be used with regex. To start, let's look at the options that can be used with it.

Syntax: `grep [options] PATTERN [file]`

Options:

`-w`	Forces `PATTERN` to match whole words only
`-x`	Forces `PATTERN` to match whole lines only
`-E`	Makes `PATTERN` an extended regular expression
`-f`	Obtains `PATTERN` from a file
`-v`	Inverts the match (prints nonmatching lines)
`-m [NUM]`	Stops after `NUM` matches
`-R`	Acts recursively when searching through directories
`—color=[WHEN]`	Displays output in color; `WHEN` is `always`, `auto`, or `never`

The hardest thing to understand when you're starting with regex is how to build a pattern to find what you're looking for. You not only need to know which option to use when calling the `grep` command, but also need to understand pattern options. Let's go through what some of these options include.

Metacharacters:

`[ ]`	Matches anything inside the brackets, either individually or as a whole, including letters or numbers. Be aware that lower- and uppercase letters are different.
`-`	Creates a range; for example, one through nine would be `1-9`. This dash can also be used to search for a range of letters such as `a-z` or `A-Z`. There are no spaces between characters when using a range.
`^`	Negates a search when used inside brackets. The caret is used outside brackets to find only lines that begin with a given string.
`$`	Similar to a caret outside brackets, finds lines based on their ending character or string.
`.`	Finds any character in its position.
`*`	Matches any character zero or more times.
`( )`	Combines multiple patterns.
`¦`	Finds left or right values, which is very useful in combination with `( )`.

These characters should be enough to get you started. The following text is used in the file_example file for all regex examples.

Step 1. Copy the following into a file called file_example:

```
My original text
Another line with the number 3

Search for the word "me"
I contain 5and6
Skip me
and me

0 + 1 = 1
Above this line is a math equation...duh!
```

Step 2. For the bracket example, look for `S[ek]` as a pattern:

```
# grep S[ek] file_example
Search for the word "me"
Skip me
```

You can see that two lines were returned from the file. The first line was returned because it contained the word *Search*, which has *Se* in it. The second line contains the word *Skip*, which matches the *Sk*.

Step 3. Try another match with `c[or]` as the pattern:

```
# grep c[on] file_example
I contain 5and6
```

Here, both letters are matched in the word *contain*. Now let's check out how range works in a pattern.

Step 4. Search for any line that contains a number using `[0-9]` as the pattern:

```
# grep [0-9] file_example
Another line with the number 3
I contain 5and6
0 + 1 = 1
```

You can see all lines with any number between zero and nine were returned. Can you think of another way you could have returned only lines that contained numbers and no words?

Step 5. Search for all letters (both lower- and uppercase) with `[A-Za-z]` as your pattern and then invert your selection:

```
# grep -v [A-Za-z] file_example
0 + 1 = 1
```

Let's run through a few more examples to finish this section.

Step 6. Search for all lines beginning with the letter *S*:

```
# grep ^S file_example
Search for the word "me"
Skip me
```

Step 7. Find any line that ends in the word *me*:

```
# grep me$ file_example
Skip me
and me
```

Step 8. Find all lines that begin with uppercase *A* or lowercase *a* and have any number of characters after it:

```
# grep ^[Aa]. file_example
Another line with the number 3
and me
Above this line is a math equation...duh!
```

As you are probably starting to realize, there is an infinite number of combinations you can use to hunt down lines or words in a file. Using regular expressions is a huge topic, and there are many great books dedicated to it. For the purposes of this book, we have covered enough for you to find what you're looking for in files. Make sure that you keep practicing with the sample and even system config files trying to find different combinations. If you're having trouble with the results of matches, I highly recommend using the `—color=always` options with your `grep` command. This highlights on your terminal what is actually being matched by your pattern, allowing you to get some better insight into what the pattern is actually looking for.

I/O Redirection

Sometimes you need to use the output from a command more than once. To accomplish this, you can redirect the output of commands using some neat command-line tricks. Let's look at the following commands:

`sort`	Sorts the output of a command or file
`wc`	Provides a word or line count
`cat`	Displays the contents of a file
`uniq`	Lists all the unique lines in a file or command output
`echo`	Outputs or displays a string
`cut`	Divides a string or output

There are also a few characters you can use to direct or redirect output of commands. These characters are

`>`	Directs output to a file or device (overwrites if the file exists)
`<`	Directs input from the file or device
`>>`	Appends output or text to a file (creates if the file doesn't exist)
`¦`	Pipes the output of one command to another
`&&`	Combines commands

This might be a lot of commands to take in at once if you've never worked with them before. After we go through a few examples, most of these commands and characters will make more sense.

Step 1. Use the `echo` command to output some text to a file:

```
# echo "This is some sample text" > file_example
```

Normally, the `echo` command just displays the text you have given it back to the screen, but because you are using the output direction character, the output is pushed to the file specified instead.

Step 2. Verify that the text was output correctly by viewing the contents of the file:

```
# cat file_example
This is some sample text
```

Having this line in a file provides a good chance to look at the `cut` command.

Syntax: `cut OPTION [FILE]`

Options:

`-d`	Specifies a delimiter
`-f`	Displays a particular field

Because you know what is in the text file you just created, why not pick it apart using the `cut` command?

Step 3. Display the third field of the text using the space as a delimiter:

```
# cut -d " " -f3 file_example
some
```

You can see that with this kind of command you can select which piece of a string or text file you'd like. Now that you know how to put some text in a file and pick it apart, see whether you can put these commands together. Using the pipe character (¦), you can combine the two previous commands to make one command pipe into another.

Step 4. Combine the two commands into a single line:

```
# cat file_example ¦ cut -d " " -f3 file_example
some
```

Here, the `cat` command normally outputs the contents of the file named file_example. However, instead, the output is sent to the `cut` command for further processing. Instead of piping the commands together, you can also use `&&` to have one command execute after another.

Step 5. Execute one command and then another:

```
# echo "This is some more text" > file_example && cut -d " " -f3
file_example
some
```

So far, you've seen some creative ways to output text to files and manipulate the text in the file. What if you output text to a file and then run the same command a second time?

Step 1. View the current contents of the sample file:

```
# cat file_example
This is some more text
```

Step 2. Output some more text to this file:

```
# echo "Different text" > file_example
```

Step 3. Verify the contents of the file again:

```
# cat file_example
Different text
```

What happened to the original text? When you use the `>`, the output is sent to a file or device. However, it always overwrites what is in the current file or device. If you want to append text, you can use the same character twice.

Step 4. Append text to the file instead of overwriting it:

```
# echo "My original text" >> file_example
```

Step 5. Verify the contents of the file:

```
# cat file_example
Different text
My original text
```

Step 6. Run the same exact command again and view the contents of the file:

```
# echo "My original text" >> file_example && cat file_example
Different text
My original text
My original text
```

Notice there are now two lines with the same text. What if this was a config file for a service with duplicate data? In that case, you can use the `uniq` command to pull only unique lines from a file, making sure that there are no duplicates.

Step 7. View only unique lines in the sample file, create a new file based on the output, and view the contents of this new file:

```
# uniq file_example > uniq_file && cat uniq_file
Different text
My original text
```

Now that you've been through a few commands and characters, you're ready to start working with the rest of the commands and some real files. Suppose you are looking through the /etc/passwd file for a particular user. When you view the contents of this file, you notice that they are not in order.

Step 1. Display the contents of /etc/passwd:

```
# cat /etc/passwd
root:x:0:0:root:/root:/bin/bash
bin:x:1:1:bin:/bin:/sbin/nologin
daemon:x:2:2:daemon:/sbin:/sbin/nologin
adm:x:3:4:adm:/var/adm:/sbin/nologin
lp:x:4:7:lp:/var/spool/lpd:/sbin/nologin
sync:x:5:0:sync:/sbin:/bin/sync
shutdown:x:6:0:shutdown:/sbin:/sbin/shutdown
halt:x:7:0:halt:/sbin:/sbin/halt
mail:x:8:12:mail:/var/spool/mail:/sbin/nologin
uucp:x:10:14:uucp:/var/spool/uucp:/sbin/nologin
operator:x:11:0:operator:/root:/sbin/nologin
games:x:12:100:games:/usr/games:/sbin/nologin
gopher:x:13:30:gopher:/var/gopher:/sbin/nologin
nobody:x:99:99:Nobody:/:/sbin/nologin
dbus:x:81:81:System message bus:/:/sbin/nologin
usbmuxd:x:113:113:usbmuxd user:/:/sbin/nologin
avahi-autoipd:x:170:170:Avahi IPv4LL Stack:/var/lib/avahi-
autoipd:/sbin/nologin
vcsa:x:69:69:virtual console memory owner:/dev:/sbin/nologin
rpc:x:32:32:Rpcbind Daemon:/var/cache/rpcbind:/sbin/nologin
rtkit:x:499:499:RealtimeKit:/proc:/sbin/nologin
abrt:x:498:498::/etc/abrt:/sbin/nologin
nscd:x:28:28:NSCD Daemon:/:/sbin/nologin
haldaemon:x:68:68:HAL daemon:/:/sbin/nologin
nslcd:x:65:55:LDAP Client User:/:/sbin/nologin
saslauth:x:497:495:"Saslauthd user":/var/empty/saslauth:/sbin/nolo-
gin
avahi:x:70:70:Avahi mDNS/DNS-SD Stack:/var/run/avahi-
daemon:/sbin/nologin
ntp:x:38:38::/etc/ntp:/sbin/nologin
rpcuser:x:29:29:RPC Service User:/var/lib/nfs:/sbin/nologin
pulse:x:496:494:PulseAudio System Daemon:/var/run/pulse:/sbin/nolo-
gin
sshd:x:74:74:Privilege-separated SSH:/var/empty/sshd:/sbin/nologin
tcpdump:x:72:72::/:/sbin/nologin
```

How are you supposed to find anything in this mess? You could do some searching with regex to find a specific user, but what if you just need a little organization in your life? If that's the case, you can use the `sort` command to make heads or tails of this mess.

Syntax: `sort [options] [FILE]`

Options:

`-b`	Ignores leading blanks
`-f`	Ignores case
`-n`	Compares according to numerical string value
`-r`	Sorts in reverse order

Step 2. Display the /etc/passwd file again, this time in a sorted format:

```
# sort /etc/passwd
abrt:x:498:498::/etc/abrt:/sbin/nologin
adm:x:3:4:adm:/var/adm:/sbin/nologin
apache:x:48:48:Apache:/var/www:/sbin/nologin
avahi-autoipd:x:170:170:Avahi IPv4LL Stack:/var/lib/avahi-
autoipd:/sbin/nologin
avahi:x:70:70:Avahi mDNS/DNS-SD Stack:/var/run/avahi-
daemon:/sbin/nologin
bin:x:1:1:bin:/bin:/sbin/nologin
daemon:x:2:2:daemon:/sbin:/sbin/nologin
dbus:x:81:81:System message bus:/:/sbin/nologin
games:x:12:100:games:/usr/games:/sbin/nologin
gdm:x:42:42::/var/lib/gdm:/sbin/nologin
gopher:x:13:30:gopher:/var/gopher:/sbin/nologin
haldaemon:x:68:68:HAL daemon:/:/sbin/nologin
halt:x:7:0:halt:/sbin:/sbin/halt
lp:x:4:7:lp:/var/spool/lpd:/sbin/nologin
mail:x:8:12:mail:/var/spool/mail:/sbin/nologin
nfsnobody:x:65534:65534:Anonymous NFS User:/var/lib/nfs:/sbin/nolo-
gin
nobody:x:99:99:Nobody:/:/sbin/nologin
nscd:x:28:28:NSCD Daemon:/:/sbin/nologin
nslcd:x:65:55:LDAP Client User:/:/sbin/nologin
ntp:x:38:38::/etc/ntp:/sbin/nologin
operator:x:11:0:operator:/root:/sbin/nologin
postfix:x:89:89::/var/spool/postfix:/sbin/nologin
pulse:x:496:494:PulseAudio System Daemon:/var/run/pulse:/sbin/nolo-
gin
root:x:0:0:root:/root:/bin/bash
rpcuser:x:29:29:RPC Service User:/var/lib/nfs:/sbin/nologin
rpc:x:32:32:Rpcbind Daemon:/var/cache/rpcbind:/sbin/nologin
rtkit:x:499:499:RealtimeKit:/proc:/sbin/nologin
saslauth:x:497:495:"Saslauthd user":/var/empty/saslauth:/sbin/nolo-
gin
shutdown:x:6:0:shutdown:/sbin:/sbin/shutdown
sshd:x:74:74:Privilege-separated SSH:/var/empty/sshd:/sbin/nologin
sync:x:5:0:sync:/sbin:/bin/sync
tcpdump:x:72:72::/:/sbin/nologin
usbmuxd:x:113:113:usbmuxd user:/:/sbin/nologin
uucp:x:10:14:uucp:/var/spool/uucp:/sbin/nologin
vcsa:x:69:69:virtual console memory owner:/dev:/sbin/nologin
```

This sorted output is much better and makes it easier to read. Why stop here, though? What if you want to know how many users are listed in this file? You could count the lines in this file, or you could use the `wc` command to do it for you.

Syntax: `wc [option] [FILE]`

Options:

`-c`	Prints the byte count
`-m`	Prints the character count
`-l`	Prints the number of lines
`-w`	Prints the word count

Step 3. Determine how many lines are in the /etc/passwd file:

```
# wc -l /etc/passwd
35 /etc/passwd
```

At this point, we have covered a bunch of commands for piping and redirection. We saw not only some simple examples, but also some practical uses. Make sure you are comfortable piping, redirecting, or combining commands for efficiency. Being able to pass the Red Hat exams requires that you have good time management skills, which you can accomplish by being able to use fewer commands more efficiently.

Compression and Archiving

One of the great things about Linux is that many things don't have a standard. This allows for various implementations and customizations among many different software programs and practices. This also can be a downfall. As you will learn when you become a system administrator, backups are the number one priority. If something should crash or become corrupt and you can't restore it because you aren't keeping up with your backups or you just don't keep any, you may be looking for a new job. Although we don't address backup programs here, this is a good lead into archiving and compression. For that task, we look at a single command in this section:

`tar`	Used for compressing and archiving files and directories

The `tar` utility has a monstrous number of options, so it would be impossible to cover them all here. Instead, let's focus on some common options used in this book.

Syntax: `tar [options] [FILE]`

Options:

`-c`	Creates a new archive
`-d`	Finds differences between the archive and file system
`-f`	Specifies the archive file to use
`-p`	Preserves files and directory permissions
`-t`	Lists the files in an archive
`-v`	Produces verbose output

Right away, you should be able to see that this little utility would be perfect to create an archive of files and directories that you could keep as a backup. Although using this tool solely as a backup strategy isn't recommended, it is useful for transferring a group of files and directories quickly. Let's look at some examples showing how to use the `tar` command.

Step 1. Create some random blank files:

```
# touch file1 file2 file3 another_file
```

Step 2. With the files in place, create a simple archive containing these files:

```
# tar cvf sample.tar file1 file2 file3 another_file
file1
file2
file3
another_file
```

Step 3. Of course, you know that you can also use some of the other tricks you learned already to make this task easier:

```
# tar cvf sample.tar <file>
another_file
file1
file2
file3
```

When an archive is created, you can also apply compression to reduce the amount of space the archive file takes up. Although multiple types of compression are supported with the use of `tar`, we look only at `gunzip` (.gz) and `bzip2` (bz2) here.

Step 4. Let's re-create the archive using the `gunzip` compression:

```
# tar cvf sample.tar.gz <file>
another_file
file1
file2
file3
```

Step 5. View the current directory to see the two current archive files:

```
# ls
sample.tar
sample.tar.gz
```

If these archives contained actual files instead of the blank ones, you'd notice that the .tar.gz file is smaller than the normal .tar file. Suppose now that you are working for a software company and the developers just sent over a build called Dec2010_MyApp.tar.gz. You need to deploy it to all your application servers, but you first need to uncompress and then extract it from the remaining .tar file. Before you deploy this build file (Dec2010_MyApp.tar.gz) to the application servers, a developer asks you to check for a file called file5.jsp within the build file. How can you tell whether this file is in the build without uncompressing and untarring the build?

Step 6. Use the `-t` option to list all the contents within the build file:

```
# tar tf Dec2010_MyApp.tar.gz
MyApp/
MyApp/build.xml
MyApp/app.config
MyApp/config/
MyApp/config/file3.sql
MyApp/config/file1.jcml
MyApp/config/file2.xml
MyApp/source/
MyApp/source/file4.jar
MyApp/source/file3.jar
MyApp/source/file5.jsp
```

As you can see, the file is indeed in the build. If this build were much larger, however (which they usually are), you could also use the `grep` command to filter out just the file you were looking for.

Step 7. Query only for the file you want within the build file:

```
# tar tf Dec2010_MyApp.tar.gz ¦ grep file5.jsp
MyApp/source/file5.jsp
```

Step 8. Query again for the same thing, but with this command, use `-v` for additional information:

```
# tar tvf Dec2010_MyApp.tar.gz ¦ grep file5.jsp
-rw-r--r-- root/root         0 2010-12-10 11:51
MyApp/source/file5.jsp
```

Step 9. Now extract this build file verbosely:

```
# tar xvf Dec2010_MyApp.tar.gz
MyApp/
MyApp/build.xml
MyApp/app.config
```

```
MyApp/config/
MyApp/config/file3.sql
MyApp/config/file1.jcml
MyApp/config/file2.xml
MyApp/source/
MyApp/source/file4.jar
MyApp/source/file3.jar
MyApp/source/file5.jsp
```

Again, you know that this build file is small, so there is no harm in allowing the extra output to see what files you're getting. You can leave out the verbose option when extracting large archives, and you get a new command prompt after the extraction is finished. You can see how the `tar` command is quite useful for many purposes and plays a large role in the real world. Make sure you're comfortable creating, extracting, and looking through archives, both compressed and uncompressed.

Summary

Think we've covered a lot? Unfortunately, this is only the tip of the iceberg when it comes to Linux. As mentioned at the beginning of this introduction, all the tools discussed here are user-level commands that you need to know to continue in this book. If you haven't seen many of the things described here, don't panic! Everyone learns at different speeds, and with some practice and repetition, you will quickly pick up everything. Remember, this book is a lab manual; hands-on is the number-one way to learn how to work with Red Hat and any other Linux operating system. Now let's move on to Chapter 1 on how to install Red Hat Enterprise Linux 6.

This chapter covers the following subjects:

- **Starting the Installation—**This section covers the installation and setup of the Red Hat Enterprise Linux operating system.
- **Verifying the Installation—**After you complete the installation, this section shows you how to verify that the installation was successful.

This chapter covers the following subjects:

- **Working with Syslog—**This section covers logging, both local and centralized.
- **Monitoring System Performance—**This section looks at how to monitor system performance.
- **Automation with cron and at—**This section covers system automation via scheduled and single-instance jobs.

The following RHCSA exam objectives are covered:

- Identify CPU and memory-intensive processes, adjust process priority with `renice`, and kill processes
- Locate and interpret system log files
- Schedule tasks using `cron`

The following RHCE exam objectives are covered:

- Produce and deliver reports on system utilization (processor, memory, disk, and network)
- Use shell scripting to automate system maintenance tasks
- Configure a system to log to a remote system
- Configure a system to accept logging from a remote system

CHAPTER 9

System Logging, Monitoring, and Automation

How do you know when something goes wrong on your system? You look at the logs, of course! In all seriousness, understanding system logging is important so that you can troubleshoot when something goes wrong. Logs are kept for almost everything that you can think of, including the kernel, boot process, and different services that are running. Aside from logs, there are also utilities you can use to monitor and check the status of your system. This chapter looks at how you can use these tools and logging to troubleshoot anything that is thrown your way. The chapter also touches on how to automate some of these tasks.

Working with Syslog

Whenever something goes wrong—and sometimes when things go right—on the system, a message is generated by the `syslog` service. These messages are used to keep track of how the system is performing, what actions are taking place, and if there is a problem with something. Red Hat comes with a `syslog` service built in and already set to start at boot. In RHEL6, the `rsyslog` package is installed by default and is the primary logging service used.

MIGRATION TIP

In RHEL5, the default logging service was split between two daemons: `syslogd` and `klogd`. `Syslogd` was used for system messages and `klogd` for kernel messages. In RHEL6, these have both been replaced with the `rsyslogd` daemon, which handles both types of messages. If you are still on RHEL5, you can install the `syslog` service via the `sysklogd` package.

The `rsyslog` service has a main config file, /etc/rsyslog.conf, that controls where messages are sent when they are generated. Because logging is a critical piece of the operating system, it is installed by default as part of the core packages.

Step 1. Just out of good habit, you should verify that the package is installed:

```
# rpm -qa | grep syslog
rsyslog-4.6.2-2.el6.x86_64
```

Because you can see that it is installed, you should also check on two other issues.

Step 2. First, make sure that the service is set to start when the system boots:

```
# chkconfig rsyslog --list
rsyslog          0:off   1:off   2:on    3:on    4:on    5:on
6:off
```

Step 3. Second, make sure that the service is currently running:

```
# service rsyslog status
rsyslogd (pid  1279) is running...
```

For RHEL05:

Step 1. Verify the installation of the package:

```
# rpm -qa | grep klog
sysklogd-1.4.1-44.el5
```

Step 2. Check that the service will start on boot:

```
# chkconfig syslog --list
syslog           0:off   1:off   2:on    3:on    4:on    5:on
6:off
```

Aside from the name difference, as we have already discussed, there are two daemons that run instead of one. They do, however, both run under a single service name.

Step 3. Verify that the service is running:

```
# service syslog status
syslogd (pid  2214) is running...
klogd (pid  2217) is running...
```

As you can see here, there are some slight differences between logging on RHEL5 and RHEL6 (such as the naming and single versus double daemons). The `rsyslog` service provides better logging features, so it is recommended to upgrade if possible.

All the messages sent to the `rsyslog` service are stored in the /var/log directory, and each message section has its own file or subdirectory. You can also define your own log files or directories. The biggest problem that arises from using any logging service is the files can become uncontrollable if left unchecked. Being able to log different alerts and to obtain information from logs, in my opinion, is an art all unto itself. There are so many ways to customize logs, and each network setup is different. You should know that there are nine different alerts that the `syslog` service uses. These alerts can be produced by different network devices and Red Hat systems.

The nine severity levels of `syslog` include the following:

- emerg
- alert
- crit
- err
- warning
- notice
- info
- debug
- none

The Config File

The /etc/rsyslog.conf file is broken down into the following parts:

Modules	Indicate whether they can be loaded or unloaded (`rsyslog` is modular)
Global directives	Specify config options that apply to the daemon (all start with a `$`)
Rules	Indicate cooperation of a selector and an action
Selector	Is based on `<facility>.<priority>`
Action	Defines what to do with messages after they're sorted by the selector

Let's look at the default config file to see some examples:

```
# cat /etc/rsyslog.conf
#rsyslog v3 config file

# if you experience problems, check
# http://www.rsyslog.com/troubleshoot for assistance

#### MODULES ####

$ModLoad imuxsock.so    # provides support for local system logging (e.g. via
logger command)
$ModLoad imklog.so      # provides kernel logging support (previously done by
rklogd)
#$ModLoad immark.so     # provides --MARK-- message capability

# Provides UDP syslog reception
#$ModLoad imudp.so
#$UDPServerRun 514

# Provides TCP syslog reception
#$ModLoad imtcp.so
#$InputTCPServerRun 514


#### GLOBAL DIRECTIVES ####

# Use default timestamp format
$ActionFileDefaultTemplate RSYSLOG_TraditionalFileFormat

# File syncing capability is disabled by default. This feature is usually not
required,
# not useful and an extreme performance hit
#$ActionFileEnableSync on


#### RULES ####

# Log all kernel messages to the console.
# Logging much else clutters up the screen.
#kern.*                                                  /dev/console

# Log anything (except mail) of level info or higher.
# Don't log private authentication messages!
*.info;mail.none;authpriv.none;cron.none                 /var/log/messages

# The authpriv file has restricted access.
authpriv.*                                               /var/log/secure

# Log all the mail messages in one place.
mail.*                                                   -/var/log/maillog
```

```
# Log cron stuff
cron.*                                                      /var/log/cron

# Everybody gets emergency messages
*.emerg                                                     *

# Save news errors of level crit and higher in a special file.
uucp,news.crit                                              /var/log/spooler

# Save boot messages also to boot.log
local7.*                                                    /var/log/boot.log

# ### begin forwarding rule ###
# The statement between the begin ... end define a SINGLE forwarding
# rule. They belong together, do NOT split them. If you create multiple
# forwarding rules, duplicate the whole block!
# Remote Logging (we use TCP for reliable delivery)
#
# An on-disk queue is created for this action. If the remote host is
# down, messages are spooled to disk and sent when it is up again.
#$WorkDirectory /var/spppl/rsyslog # where to place spool files
#$ActionQueueFileName fwdRule1 # unique name prefix for spool files
#$ActionQueueMaxDiskSpace 1g   # 1gb space limit (use as much as possible)
#$ActionQueueSaveOnShutdown on # save messages to disk on shutdown
#$ActionQueueType LinkedList   # run asynchronously
#$ActionResumeRetryCount -1    # infinite retries if host is down
# remote host is: name/ip:port, e.g. 192.168.0.1:514, port optional
#*.* @@remote-host:514
# ### end of the forwarding rule ###
```

You can see from the output where some of the files are stored. Also shown is a sample rule.

Log Rotation

If left alone, logs can grow to enormous size. Luckily for you, the `logrotate` command allows you to rotate logs before they become too big. By default, the `logrotate` command is called daily to cycle the log files into a new file. The parameters of the command are defined in its config file, /etc/logrotate.conf, and the /etc/logrotate.d directory. However, the command itself is called from /etc/cron.daily/logrotate, a `cron` job that is run daily (as already mentioned). You can always call the `logrotate` command yourself if you'd like to rotate your logs, but don't do this too frequently because you'll end up with tons of rotated logs and you'll be right back to square one with a mess of unorganized log files.

Syntax: `logrotate [options] configfile`

Options:

`-d`	Debugs; doesn't do anything but test
`-f`	Forces file rotation
`-v`	Provides verbose output

The following example rotates the current logs as defined in the config file:

```
# logrotate /etc/logrotate.conf
```

Centralized Logging

If you'd like to set up a centralized syslog server, you need to choose which server you'd like to store all your logs on. For this example, you can use the RHEL01 system as the centralized server and configure it to receive logs from the other systems on the network.

Step 1. Look at a section of the /etc/rsyslog.conf file again:

```
#### MODULES ####

$ModLoad imuxsock.so    # provides support for local system log-
ging (e.g. via logger command)
$ModLoad imklog.so      # provides kernel logging support (previ-
ously done by rklogd)
#$ModLoad immark.so     # provides --MARK-- message capability

# Provides UDP syslog reception
#$ModLoad imudp.so
#$UDPServerRun 514

# Provides TCP syslog reception
#$ModLoad imtcp.so
#$InputTCPServerRun 514
```

In the modules section, you can see two areas that will provide the capability to receive logs from other systems. The standard is to use the UDP protocol on port 514. By uncommenting the section related to UDP reception, you essentially can provide a way for the syslog server to become the centralized server for the network.

Step 2. Uncomment the following two lines:

```
$ModLoad imudp.so
$UDPServerRun 514
```

Step 3. When you're finished, you need to save the file, exit, and restart the `syslog` service:

```
# service rsyslog restart
Shutting down system logger:                               [  OK  ]
Starting system logger:                                    [  OK  ]
```

As with any service that you'd like to make available to your network, you need to create a firewall allowing UDP 514 to be opened up.

Step 4. Use `iptables` to create the required firewall rule:

```
# iptables -I INPUT 5 -p udp -m udp --dport 514 -j ACCEPT
```

Step 5. Save the firewall rule you just created:

```
# service iptables save
Saving firewall rules to /etc/sysconfig/iptables:          [  OK  ]
```

Step 6. Then restart the `iptables` service:

```
# service iptables restart
iptables: Flushing firewall rules:                         [  OK  ]
iptables: Setting chains to policy ACCEPT: filter          [  OK  ]
iptables: Unloading modules:                               [  OK  ]
iptables: Applying firewall rules:                         [  OK  ]
```

The firewall rules might not make sense at the moment because we haven't covered `iptables` or firewall rules yet. After you read Chapter 12, "System Security," return here to make sure you understand what is happening in the creation of firewall rules.

MIGRATION TIP

Unfortunately, due to the revamp of the `syslog` service, centralized logging for RHEL5 is quite different. On RHEL5, the `syslog` config file doesn't contain a modules section to enable remote management. Instead, you need to edit the daemon config file. We cover centralized logging for RHEL5 in the next section.

Now that the security requirements have been addressed and the centralized server is configured, you can shift your attention to the client systems. Let's look at RHEL02 for the client system. You can config RHEL02 to send some or all of its log files to the centralized syslog server (RHEL01).

You can use the following syntax when configuring log files to be sent to the centralized server:

```
<log file>           @<hostname or IP of system (local or remote)>
```

You can edit the RHEL02 system to point all security logs to the central server:

```
# nano /etc/rsyslog.conf
authpriv.*           @172.168.1.1
```

Save the file and restart the `syslog` service on RHEL02:

```
# service rsyslog restart
Shutting down system logger:                              [  OK  ]
Starting system logger:                                   [  OK  ]
```

If you log out and log back in on RHEL02, it generates a security event. You can then jump over to the RHEL01 host and check the logs to see the event generated by the client system RHEL02.

REAL-WORLD TIP

When you're pointing logs to a centralized server, either local or remote, there are a few special options you can use:

```
# Send messages using the TCP protocol instead of UDP
Authpriv.*                        @@172.168.1.1
# Displays messages on the console instead of sending them remotely
Authpriv.*                        @system
# Discard all messages generated for this log file
Authpriv.*                        ~
# Send alert to all log files ** use with caution **
Authpriv.*                        *
```

These options can be used in conjunction with forwarding your logs to keep a local copy on your server.

Centralized Logging (The RHEL5 Way)

To set up centralized logging for RHEL5, do the following:

Step 1. Edit the /etc/sysconfig/syslog file to include the `-r` option:

```
# nano /etc/sysconfig/syslog
# Options to syslogd
# -m 0 disables 'MARK' messages.
# -r enables logging from remote machines
# -x disables DNS lookups on messages received with -r
# See syslogd(8) for more details
SYSLOGD_OPTIONS="-m 0 -r"
```

Step 2. Save the file, exit, and restart the `syslog` service:

```
# service syslog restart
Shutting down kernel logger:                              [  OK  ]
Shutting down system logger:                              [  OK  ]
Starting system logger:                                   [  OK  ]
Starting kernel logger:                                   [  OK  ]
```

With the `syslog` service now accepting remote logs, you need to create a firewall rule. This process is the same as on RHEL6. One last modification that needs to be made on RHEL5 is the adjustment of some SELinux protections.

Step 3. Query the required Boolean values:

```
# getsebool -a | grep logd
klogd_disable_trans --> off
syslogd_disable_trans --> off
```

Step 4. Change the values to allow incoming logs:

```
# setsebool -P klogd_disable_trans=1 syslogd_disable_trans=1
```

Step 5. Verify the preceding Booleans have been changed:

```
# getsebool -a | grep logd
klogd_disable_trans --> on
syslogd_disable_trans --> on
```

Although we haven't covered SELinux yet (it is covered in Chapter 11, "SELinux"), the Boolean values are required here if you are still using RHEL5. After you read through Chapter 11, you can refer back here to make sure you understand the changes being made.

User Login Events

Aside from the normal logs generated and used by the `syslog` service, there are two special commands that deal with system logins. These two commands have special logs that can be read only through the use of these commands:

`lastlog`	Lists login records
`faillog`	Lists failed login attempts

You can use these two commands for viewing login events. These two commands are useful for hunting down attacks where users are trying to force themselves into other users' accounts (called *a brute-force attack*).

Syntax: `lastlog [options]`

Options:

`-b DAYS`	Displays results older than `DAYS`
`-u LOGIN`	Displays results for the user `LOGIN`

Show the last time the user01 account logged in:

```
# lastlog -u user01
Username         Port     From             Latest
user01           tty                        Fri Sep 10 05:16:42 -0400 2010
```

You can also view more details using the `faillog` command.

Syntax: `faillog [options]`

Options:

`-a`	Displays all events
`-l SEC`	Locks the account for `SEC` seconds after a failed login
`-u LOGIN`	Prints records for user `LOGIN`

Show more information about the user01 account login events:

```
# faillog -u user01
Login       Failures Maximum Latest                     On
user01          0        0
```

The ability to work with `syslog` is important to being able to help troubleshoot any issue on Linux, not just Red Hat. When you're trying to find an issue or resolve one, the log files should always be your first stop.

Monitoring System Performance

Every time a program or command is run, a process is created for it. These processes are all unique and identified by the process identification (PID) that becomes allocated to it. System processes are critical to keeping the system up and running or providing services to clients. Management of processes can help keep the system stable or help when the system becomes unstable. Here are some of the process management commands you can use:

`ps`	Displays information about running processes
`kill`	Terminates a process
`pgrep`	Finds a process based on its PID
`pidof`	Displays all processes related to a service or command
`top`	Monitors system resources (similar to Task Manager in Windows)
`renice`	Adjusts the priority of a particular process

First, let's look at the processes the root user currently owns by using the ps command:

```
# ps
  PID TTY          TIME CMD
 4474 pts/3    00:00:00 bash
 4506 pts/3    00:00:00 ps
```

The ps command includes a number of options for producing different types of output, including viewing all processes currently running on the system.

To view processes with more detailed information, you can use the following command:

```
# ps u
USER     PID %CPU %MEM    VSZ   RSS TTY   STAT START   TIME COMMAND
root    2387  0.0  0.1   1660   424 tty1  Ss+  07:30   0:00 /sbin/mingetty tty1
root    2388  0.0  0.1   1660   420 tty2  Ss+  07:30   0:00 /sbin/mingetty tty2
root    2389  0.0  0.1   1660   420 tty3  Ss+  07:30   0:00 /sbin/mingetty tty3
root    2390  0.0  0.1   1660   420 tty4  Ss+  07:30   0:00 /sbin/mingetty tty4
root    2391  0.0  0.1   1660   448 tty5  Ss+  07:30   0:00 /sbin/mingetty tty5
root    2392  0.0  0.1   1660   420 tty6  Ss+  07:30   0:00 /sbin/mingetty tty6
```

Here, you see not only the PID, but also the CPU and memory utilization. A common set of options that I use when working with the ps command is aux combined with the grep command to weed out anything I'm not interested in. This way, I can get detailed information about a particular process. Say you'd like to see what information is available about any process pertaining to the SSH service:

```
# ps aux | grep ssh
root       4286  0.0  0.1  62616  1216 ?    Ss   Sep23  0:00 /usr/sbin/sshd
root      15872  0.0  0.3  90116  3248 ?    Ss   10:06  0:00 sshd: user01 [priv]
user01   15874  0.0  0.1  90116  1740 ?    S    10:06  0:00 sshd: user01@pts/0
user01   15921  0.0  0.0  61176   728 pts/0  R+  10:14  0:00 grep ssh
```

You also could use

```
# ps aux
```

With this command, however, you would receive 59 lines of output that you would have had to go through to view those 4 lines that you actually need. You can see how narrowing down your output can be really helpful.

Let's also look at some of the other options the ps command offers:

```
$ ps —help
********* simple selection *********  ********* selection by list *********
-A all processes                        -C by command name
```

```
-N negate selection                    -G by real group ID (supports names)
-a all w/ tty except session leaders  -U by real user ID (supports names)
-d all except session leaders         -g by session OR by effective group name
-e all processes                      -p by process ID
T  all processes on this terminal     -s processes in the sessions given
a  all w/ tty, including other users  -t by tty
g  OBSOLETE — DO NOT USE              -u by effective user ID (supports names)
r  only running processes             U  processes for specified users
x  processes w/o controlling ttys     t  by tty
*********** output format ********** *********** long options ***********
-o,o user-defined  -f full            --Group --User --pid --cols --ppid
-j,j job control   s  signal          --group --user --sid --rows --info
-O,O preloaded -o  v  virtual memory  --cumulative --format --deselect
-l,l long          u  user-oriented   --sort --tty --forest --version
-F   extra full    X  registers       --heading --no-heading --context
                    ********* misc options *********
-V,V  show version      L  list format codes  f  ASCII art forest
-m,m,-L,-T,H  threads   S  children in sum    -y change -l format
-M,Z  security data     c  true command name  -c scheduling class
-w,w  wide output       n  numeric WCHAN,UID  -H process hierarchy
```

You can see here that there are numerous output formats and sort features. Knowing how to manipulate the output of the ps command and find what you're looking for really helps when troubleshooting on the exams, but it also plays a huge role in troubleshooting in the real world.

What happens if one of the processes running on your system becomes out of control? In that case, you can use the `kill` command to terminate the process, even if it isn't responding (also called a *runaway process*). To use the `kill` command, however, you need to know the PID of the process that you want to kill (are you starting to see why the ps command is so helpful?).

Syntax: `kill PID`

If you want to stop the SSH service because it isn't responding, you just have to look for the PID associated with the SSH daemon. If you look back at the output, you can see this is PID 4286. To kill the process forcefully and effectively stop the SSH service, you can do the following:

```
# kill 4286
```

Sometimes if the `kill` command doesn't work the way you intended it to, you can also call it with the `-9` option to give it priority on the system:

```
# kill -9 4286
```

What happens if you don't know the PID of the process you want to terminate? How could you look that up if you weren't sure what to query from the output of the `ps` command? There are actually two other commands you can use to determine the PID(s) of a service or command: `pidof` and `pgrep`.

First, let's look at `pidof`, to which you can pass just the name of the service or daemon. To find the PID(s) belonging to the SSH service, use the following:

```
# pidof sshd
15874 15872 4286
```

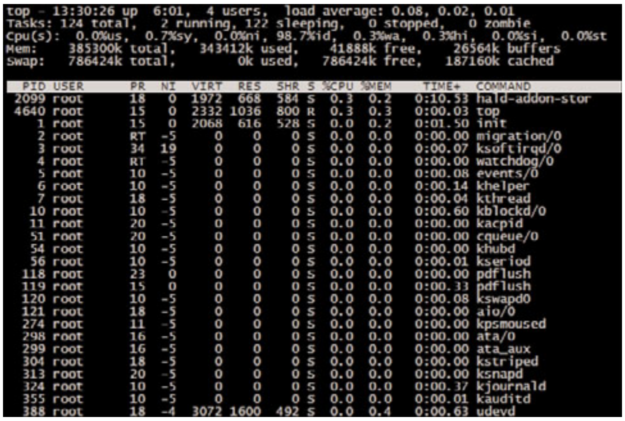
To achieve the same information in an easier-to-read format, you can use the `pgrep` command:

```
# pgrep sshd
4286
15872
15874
```

It's good to know that you can hunt down and kill processes or even just find out how many are running, but what if you need more information? Suppose you need to know how much of the CPU a particular process is taking up? Let's look at the final command: `top`. This command gives you an overview of processes on the system, including memory usage, CPU utilization, and more.

To check out the system overall, issue the following command and you get results similar to those shown in Figure 9-1:

```
#top
```



```
top - 13:30:26 up  6:01,  4 users,  load average: 0.08, 0.02, 0.01
Tasks: 124 total,   2 running, 122 sleeping,   0 stopped,   0 zombie
Cpu(s):  0.0%us,  0.7%sy,  0.0%ni, 98.7%id,  0.3%wa,  0.3%hi,  0.0%si,  0.0%st
Mem:    385300k total,   343412k used,    41888k free,    26564k buffers
Swap:   786424k total,        0k used,   786424k free,   187160k cached

  PID USER      PR  NI  VIRT  RES  SHR S %CPU %MEM    TIME+  COMMAND
 2099 root      18   0  1972  668  584 S  0.3  0.2   0:10.53 hald-addon-stor
 4640 root      15   0  2332 1036  800 R  0.3  0.3   0:00.03 top
    1 root      15   0  2068  616  528 S  0.0  0.2   0:01.50 init
    2 root      RT  -5     0    0    0 S  0.0  0.0   0:00.00 migration/0
    3 root      34  19     0    0    0 S  0.0  0.0   0:00.07 ksoftirqd/0
    4 root      RT  -5     0    0    0 S  0.0  0.0   0:00.00 watchdog/0
    5 root      10  -5     0    0    0 S  0.0  0.0   0:00.08 events/0
    6 root      10  -5     0    0    0 S  0.0  0.0   0:00.14 khelper
    7 root      18  -5     0    0    0 S  0.0  0.0   0:00.04 kthread
   10 root      10  -5     0    0    0 S  0.0  0.0   0:00.60 kblockd/0
   11 root      20  -5     0    0    0 S  0.0  0.0   0:00.00 kacpid
   51 root      20  -5     0    0    0 S  0.0  0.0   0:00.00 cqueue/0
   54 root      10  -5     0    0    0 S  0.0  0.0   0:00.00 khubd
   56 root      10  -5     0    0    0 S  0.0  0.0   0:00.01 kseriod
  118 root      23   0     0    0    0 S  0.0  0.0   0:00.00 pdflush
  119 root      15   0     0    0    0 S  0.0  0.0   0:00.33 pdflush
  120 root      10  -5     0    0    0 S  0.0  0.0   0:00.08 kswapd0
  121 root      18  -5     0    0    0 S  0.0  0.0   0:00.00 aio/0
  274 root      11  -5     0    0    0 S  0.0  0.0   0:00.00 kpsmoused
  298 root      16  -5     0    0    0 S  0.0  0.0   0:00.00 ata/0
  299 root      16  -5     0    0    0 S  0.0  0.0   0:00.00 ata_aux
  304 root      18  -5     0    0    0 S  0.0  0.0   0:00.00 kstriped
  313 root      20  -5     0    0    0 S  0.0  0.0   0:00.00 ksnapd
  324 root      10  -5     0    0    0 S  0.0  0.0   0:00.37 kjournald
  355 root      10  -5     0    0    0 S  0.0  0.0   0:00.01 kauditd
  388 root      18  -4  3072 1600  492 S  0.0  0.4   0:00.63 udevd
```

Figure 9-1 Results of running **`top`**.

The output in Figure 9-1 shows real-time information about the system resources. You can use the `q` to quit from the `top` command.

When you're comfortable working with processes, you can then make some more advanced adjustments, such as changing the priority of a particular process. Let's say that you have a custom application running on your system. You can use the `renice` command to give that particular process higher priority on the CPU.

Syntax: `renice <priority> [options]`

Options:

`-p PID`	Changes process priority for a particular PID
`-u user`	Changes process priority for a particular user(s)

The priority values range from –20 (first priority) to 20 (dead last priority). Only the root user may set processes to use a priority under 0. Going back to the example, you can give the application extra priority by using the following:

```
# renice -2 3874
```

This command changes the application process (3874) to have better priority than its default priority of 0. Knowing how to work with system processes and extract information from your system will help you become a better system administrator. It can also help you troubleshoot faster on the Red Hat exams.

In the next section, we look at job scheduling, which can be combined with some of the commands just covered to automate system monitoring.

Automation with cron and at

In any operating system, it is possible to create jobs that you want to reoccur. This process, known as *job scheduling*, is usually done based on user-defined jobs. For Red Hat, this process is handled by the `cron` service, which can be used to schedule tasks (also called *jobs*). By default, Red Hat comes with a set of predefined jobs that occur on the system (hourly, daily, weekly, monthly, and with arbitrary periodicity). As an administrator, however, you can define your own jobs and allow your users to create them as well.

Step 1. Although the `cron` package is installed by default, check to make sure it is installed on your system:

```
# rpm -qa | grep cron
cronie-1.4.4-2.el6.x86_64
cronie-anacron-1.4.4-2.el6.x86_64
crontabs-1.10-32.1.el6.noarch
```

Step 2. If, for some reason, the package isn't installed, install it now:

```
# yum install -y cronie cronie-anacron crontabs
```

MIGRATION TIP
In RHEL5, the `cron` package was called `vixie-cron`, which has been replaced in RHEL6 with `cronie`.

Step 3. Verify that the `cron` service is currently running:

```
# service crond status
crond (pid  2239) is running...
```

Step 4. Also verify that the service is set to start when the system boots:

```
# chkconfig --list crond
crond             0:off   1:off   2:on    3:on    4:on    5:on
6:off
```

To start working with `cron`, you first need to look at the two config files that control access to the `cron` service. These two files are:

/etc/cron.allow

/etc/cron.deny

The /etc/cron.allow file:

- If it exists, only these users are allowed (cron.deny is ignored).
- If it doesn't exist, all users except cron.deny are permitted.

The /etc/cron.deny file:

- If it exists and is empty, all users are allowed (Red Hat Default).

For both files:

- If neither file exists, root only.

EXAM TIP
The rules surrounding these two files can be kind of confusing at first, so you might want to create some sample jobs to gain a better understanding of how changing these two files affects the permissions of the `cron` service.

Creating cron Jobs

The default setting for Red Hat allows any user to create a `cron` job. As the root user, you also have the ability to edit and remove any `cron` job you want. Let's jump into creating a `cron` job for the system. You can use the `crontab` command to create, edit, and delete jobs.

Syntax: `crontab [-u user] [option]`

Options:

`-e`	Edits the user's `crontab`
`-l`	Lists the user's `crontab`
`-r`	Deletes the user's `crontab`
`-i`	Prompts before deleting the user's `crontab`

Before you start using the `crontab` command, however, you should look over the format it uses so you understand how to create and edit `cron` jobs. Each user has her own `crontab` file in /var/spool/cron (the file for each user is created only after the user creates her first `cron` job), based on the username of each user. Any "allow" actions taken by the `cron` service are logged to /var/log/cron.

View the /etc/crontab file to understand its syntax:

```
# grep ^# /etc/crontab
# For details see man 4 crontabs
# Example of job definition:
# .---------------- minute (0 - 59)
# |  .------------- hour (0 - 23)
# |  |  .---------- day of month (1 - 31)
# |  |  |  .------- month (1 - 12) OR jan,feb,mar,apr ...
# |  |  |  |  .---- day of week (0 - 6) (Sunday=0 or 7) OR sun,mon,tue,etc.
# |  |  |  |  |
# *  *  *  *  *  command to be executed
```

This file clearly spells out the values that can exist in each field. You must make sure that you provide a value for each field; otherwise, the `crontab` will not be created. You can also define step values by using `*/<number to step by>`. For example, you could put `*/5` in the minute field to mean every fifth minute.

The best way to understand these fields is to create a `crontab` file and make some sample jobs. Using a text editor, create the following file in the /tmp directory.

Sample script for `cron`:

```
# nano /tmp/sample_script
#!/bin/bash
#
# Send a msg to all users on the console
#
wall "Hello World"
```

Save the file and set the following permissions:

```
# chmod 775 /tmp/sample_script
```

Now create a `cron` job to launch the sample script. Because you are using the root user account, you can create a `crontab` for the normal user account user01.

Step 1. Set up user01's `crontab`:

```
# crontab -u user01 -e
```

Step 2. Add the following line:

```
* * * * * /tmp/sample_script
```

Step 3. Save the file and quit the editor.

Because you are using `*` in every field to test, in about 60 seconds you will see the script execute, and it should display `"`Hello World`"` on your screen.

EXAM TIP
The `crontab` file uses the `vi` editor to make changes to itself. If you aren't familiar with this editor, be aware that the input commands are slightly different from that of `nano` or `emacs`. Make sure you know how to use the `vi` editor for the Red Hat exams.

Obviously, you don't want messages every 60 seconds on your system, but you get the idea of how `cron` and `crontab` should work. Let's list the current jobs that user01 has set up, and you should see the job just created (do this just for verification purposes).

Step 4. List the current `cron` jobs of user01:

```
# crontab -u user01 -l
* * * * * /tmp/sample_script
```

You can now edit the `crontab` again to remove the single line, effectively deleting that individual job, or you can just delete the user's `crontab` entirely.

Step 5. To remove a user's `crontab` jobs, use the following command:

```
# crontab -u user01 -r
```

Step 6. You can verify the activity on different `crontabs` by...wait for it...looking at the log files!

```
# tail /var/log/cron
Sep 10 09:08:01 new-host crond[4213]: (user01) CMD
(/tmp/sample_script)
Sep 10 09:08:38 new-host crontab[4220]: (root) LIST (user01)
Sep 10 09:09:01 new-host crond[4224]: (user01) CMD
(/tmp/sample_script)
Sep 10 09:10:01 new-host crond[4230]: (user01) CMD
(/tmp/sample_script)
Sep 10 09:11:01 new-host crond[4236]: (user01) CMD
(/tmp/sample_script)
Sep 10 09:12:01 new-host crond[4242]: (user01) CMD
(/tmp/sample_script)
Sep 10 09:13:01 new-host crond[4248]: (user01) CMD
(/tmp/sample_script)
Sep 10 09:13:06 new-host crontab[4251]: (root) LIST (user01)
Sep 10 09:14:01 new-host crond[4253]: (user01) CMD
(/tmp/sample_script)
Sep 10 09:14:15 new-host crontab[4258]: (root) DELETE (user01)
```

You can see that the `cron` service is executing the /tmp/sample_script file, and you can see the action after you deleted it. The process of creating `crontabs` and scheduling jobs is the same for all users on the system, including the root user.

MIGRATION TIP

In RHEL5, the /etc/crontab file was used as the root user's `crontab`. This `crontab` followed a different format from that of all other system users. The /etc/crontab file has an extra field between the last time field and command field. In this extra field, you must specify which user you want the job to run as.

Here is what the /etc/crontab file looks like for RHEL5 systems:

```
# cat /etc/crontab
SHELL=/bin/bash
PATH=/sbin:/bin:/usr/sbin:/usr/bin
MAILTO=root
HOME=/

# run-parts
01 * * * * root run-parts /etc/cron.hourly
02 4 * * * root run-parts /etc/cron.daily
22 4 * * 0 root run-parts /etc/cron.weekly
42 4 1 * * root run-parts /etc/cron.monthly
```

Here, the root user is defined to run the specified commands. This user field is now optional in RHEL6.

What do you think happens if you set up `cron` jobs to run during the night (say, to run some reports) and you shut down the system right before you go home? Well, it turns out that there is another great feature of `cron`. The /etc/anacrontab file defines jobs that should be run every time the system is started. If your system is turned off during the time that a `cron` job should have run, when the system boots again, the `cron` service will call /etc/anacrontab to make sure that all missed `cron` jobs are run.

Let's look at the /etc/anacrontab file:

```
# cat anacrontab
SHELL=/bin/sh
PATH=/sbin:/bin:/usr/sbin:/usr/bin
MAILTO=root
# the maximal random delay added to the base delay of the jobs
RANDOM_DELAY=45
# the jobs will be started during the following hours only
START_HOURS_RANGE=3-22

#period in days   delay in minutes   job-identifier   command
1        5        cron.daily              nice run-parts /etc/cron.daily
7        25       cron.weekly             nice run-parts /etc/cron.weekly
@monthly 45       cron.monthly            nice run-parts /etc/cron.monthly
```

The comments in this file make it easy to understand how jobs are defined. If your system is constantly on and you don't require `anacron` to be run, you can uninstall the `cronie-anacron` package, which controls just the `anacron` commands for the `cron` service.

MIGRATION TIP

In RHEL5, `anacron` was a separate service from `cron`. The `anacron` service alone was called when the system booted and similarly ran the /etc/anacrontab file. This file looks similar to the RHEL6 version without any comments:

```
# cat anacrontab
SHELL=/bin/sh
PATH=/sbin:/bin:/usr/sbin:/usr/bin
MAILTO=root

1       65      cron.daily              run-parts /etc/cron.daily
7       70      cron.weekly             run-parts /etc/cron.weekly
30      75      cron.monthly            run-parts /etc/cron.monthly
```

For RHEL5, you need to ensure that this service is set to run when the system boots in order for it to be effective:

```
# chkconfig anacron –list
anacron         0:off   1:off   2:on    3:on    4:on    5:on    6:off
```

MIGRATION TIP

In RHEL5, you also need to disable SELinux protection for the `cron` service to function properly. Although we haven't covered SELinux yet, you can use the following steps to configure it:

Step 1. Query the required Boolean values:

```
# getsebool -a | grep crond
crond_disable_trans --> off
```

Step 2. Change the values to allow incoming logs:

```
# setsebool -P crond_disable_trans=1
```

Step 3. Verify the preceding Booleans have been changed:

```
# getsebool -a | grep crond
crond_disable_trans --> on
```

Single Jobs with at

Although you can use the `cron` service to schedule jobs that you want to occur more than once, you can use a service called `atd` for single-instance jobs.

Step 1. As with any service thus far, you need to verify that the package is installed:

```
# rpm -qa | grep ^at
at-3.1.10-42.el6.x86_64
```

Step 2. Check that the service is currently running:

```
# service atd status
atd (pid  2300) is running...
```

Step 3. Finally, verify that the service is set to start on system boot:

```
# chkconfig --list atd
atd             0:off   1:off   2:off   3:on    4:on    5:on
6:off
```

The `atd` service uses two files in the same manner that `cron` does to control access to the service. These files are

/etc/at.allow

/etc/at.deny

The /etc/at.allow file:

- If it exists, only these users are allowed (at.deny is ignored).
- If it doesn't exist, all users except at.deny are permitted.

The /etc/at.deny file:

- If it exists and is empty, all users are allowed (Red Hat default).

For both files:

- If neither file exists, root only.

The `atd` service also includes a single command, `at`, that is used to set up the jobs you want to run.

Syntax: `at [options]`

Options:

`-l`	Lists all jobs in the queue
`-d ID`	Removes a job from the queue
`-m`	Sends mail to the user when the job is complete
`-f FILE`	Reads input from the file
`-v`	Shows the time the job will be executed

The `at` command enables you to specify a time in many different formats, making it really flexible. Let's look at a few examples of the time formats you can use:

```
# at 9am
# at now + 3 days
# at 1:30 3/22/10
```

After you specify a time, your command prompt changes:

```
# at 10:07am
at>
```

Now you just need to enter any commands that you want to execute for your job at the specified time. You can finish by pressing Ctrl+D to end the command input and send the job to the queue:

```
# at 10:07am
at>
at> wall "Hello World"
at> <EOT>
```

Now that a job is queued, let's view what the file that holds this job looks like.

Step 1. Query the /var/spool/at directory for jobs:

```
# ls /var/spool/at
a000010147997f  spool
```

Unlike with the `cron` service, the names of the `at` jobs aren't really meaningful. Instead of viewing the directory that holds the `at` jobs, you might find it easier to list the jobs waiting to be run like you did with the `cron` service.

Step 2. View the currently queued jobs using the `at` command:

```
# at -l
1       2010-10-27 10:07 a root
```

Step 3. You can also use the `atq` command for the exact same results:

```
# atq
1       2010-10-27 10:07 a root
```

You can see one job currently in the queue was sent by the root user and is waiting to run at 10:07 a.m. Instead of typing out all the jobs you might want to run, you could also use the `-f` option to specify a file containing a list of commands that you want executed instead:

```
# at -f cmds_file 11pm
```

When you put jobs in the queue, notice that they are given ID numbers. This information is important in case there is a job that you want to delete from the queue.

Step 4. Add a temporary job to the queue:

```
# cd ~
# touch test_job && chmod 775 test_job
# at -f test_job 11pm
```

Step 5. Verify that the job made it to the queue:

```
# atq
1       2010-12-04 23:00 a root
```

Step 6. Delete the job from the queue:

```
# at -d 1
```

Step 7. You can also use the `atrm` command to achieve the same results:

```
# atrm 1
```

Step 8. Verify that the job is truly gone:

```
# atq
```

You should now be able to schedule single jobs and reoccurring jobs. Job management is important in making sure that your system runs smoothly. It also helps when you're automating tasks so you don't have to do them every day.

Summary

Knowing how to work with logs and where to look for information is critical when you're troubleshooting the system. A centralized syslog server helps in many ways with making log collection and management easier. You can also use tools such as `ps` and `top` to look into resource issues that the system may be having. In this chapter, we looked at the `cron` and `atd` services, each of which can be used for scheduling jobs. Using these tools, you can help track down problems that arise and look for trends of reoccurring issues. In the next chapter, we look at the kernel, the brains behind the Red Hat operating system.

Review Questions

1. What option can you change in the `rsyslog` config file to accept remote logs (acting as a centralized logging server)?
2. What two commands are special for dealing with user login events?
3. Can you name the two commands that can be used to view the free space on the system?
4. What command can you use to view system processes and their CPU usage?
5. The `at` command is used to schedule reoccurring system jobs. True or False?
6. What happens to jobs that are scheduled to run while the system is off?
7. What command can be used to view the queue for `at` service jobs?
8. What is the `top` command used for?

Answers to Review Questions

1. Uncomment the following line in the /etc/rsyslog.conf file:

   ```
   #$ModLoad imudp.so
   #$UDPServerRun 514
   ```

2. The `lastlog` and `faillog` commands are used to view user login–related events?
3. The `du` and `df` commands are used to view available space on the system.
4. Use the `ps` command to view processes and their CPU usage.
5. False. The `at` command is used to schedule one-time-only jobs. The `cron` service handles reoccurring system jobs.
6. When the system starts up again, the `cron` service will run any jobs that were missed while the system was off. On Red Hat Enterprise Linux 5, the `anacron` service handles this functionality.
7. `atq`
8. Use the `top` command to view CPU and memory usage.

Lab 9

Task 1 – Centralized Logging

Step 1. Set up a centralized log server on RHEL02 using the following guidelines:
- **a.** The server must accept logs from all clients on the 172.168.1.0/24 subnet.
- **b.** All security logs should remain local to the system that they are on.
- **c.** Only /var/log/messages and /var/log/dmesg should be forwarded to the centralized server.

Step 2. Set up each server or client system in the lab to forward the appropriate logs to RHEL02.

The task is complete when all logs are set up correctly and the centralized log server (RHEL02) is collecting the correct logs.

Task 2 – System Monitoring

Step 1. Create a BASH script on RHEL01 using the following guidelines:

- **a.** The script must output the `ps aux` command to a file called system_mon located in the /opt directory.
- **b.** The script must output the amount of free memory to a file called system_mon located in the /opt directory.
- **c.** The script must append the date to the system_mon file.

The task is complete when the script records the necessary information in the required file and the file is named correctly.

Task 3 – Scheduling Jobs

Perform the following tasks on RHEL01:

Step 1. Set up a `cron` job to execute every day at noon. The job should execute the script that you created in Task 2.

Step 2. Create a single job to occur 10 minutes from now that will output the date and time to all users who are logged in to the system.

The task is complete when all jobs have been scheduled and they execute successfully.

Index