



redhat®

**Red Hat System
Administration I
Student Workbook**

RH124-RHEL7-en-1-20140606
MAN-RH124SKE-R2

**RED HAT®
TRAINING**



Comprehensive, hands-on training that solves real world problems

Red Hat System Administration I

Student Workbook

RED HAT SYSTEM ADMINISTRATION I

Red Hat Enterprise Linux 7 RH124

Red Hat System Administration I

Edition 1

Authors: Susan Lauber, Philip Sweany, Rudolf Kastl, George Hacker
Editor: Steven Bonneville

Copyright © 2014 Red Hat, Inc.

The contents of this course and all its modules and related materials, including handouts to audience members, are Copyright © 2014 Red Hat, Inc.

No part of this publication may be stored in a retrieval system, transmitted or reproduced in any way, including, but not limited to, photocopy, photograph, magnetic, electronic or other record, without the prior written permission of Red Hat, Inc.

This instructional program, including all material provided herein, is supplied without any guarantees from Red Hat, Inc. Red Hat, Inc. assumes no liability for damages or legal action arising from the use or misuse of contents or details contained herein.

If you believe Red Hat training materials are being used, copied, or otherwise improperly distributed please e-mail training@redhat.com or phone toll-free (USA) +1 (866) 626-2994 or +1 (919) 754-3700.

Red Hat, Red Hat Enterprise Linux, the Shadowman logo, JBoss, Hibernate, Fedora, the Infinity Logo, and RHCE are trademarks of Red Hat, Inc., registered in the United States and other countries.

Linux® is the registered trademark of Linus Torvalds in the United States and other countries.

Java® is a registered trademark of Oracle and/or its affiliates.

XFS® is a registered trademark of Silicon Graphics International Corp. or its subsidiaries in the United States and/or other countries.

The OpenStack® Word Mark and OpenStack Logo are either registered trademarks/service marks or trademarks/service marks of the OpenStack Foundation, in the United States and other countries and are used with the OpenStack Foundation's permission. We are not affiliated with, endorsed or sponsored by the OpenStack Foundation, or the OpenStack community.

All other trademarks are the property of their respective owners.

Contributors: Rob Locke, Bowe Strickland, Scott McBrien, Wander Boessenkool, Forrest Taylor

Reviewers: Michael Phillips, David Bucknell, Aaron Hicks, Jay Ramsurrun

Document Conventions	ix
Notes and Warnings	ix
Introduction	xi
Red Hat System Administration I	xi
Orientation to the Classroom Environment	xii
Internationalization	xiii
1. Accessing the Command Line	1
Accessing the Command Line Using the Local Console	2
Practice: Local Console Access Terms	5
Accessing the Command Line Using the Desktop	8
Practice: The GNOME 3 Desktop Environment	13
Executing Commands Using the Bash Shell	15
Practice: Bash Commands and Keyboard Shortcuts	19
Lab: Accessing the Command Line	22
2. Managing Files From the Command Line	27
The Linux File System Hierarchy	28
Practice: File System Hierarchy	31
Locating Files by Name	34
Practice: Locating Files and Directories	39
Managing Files Using Command-Line Tools	41
Practice: Command-Line File Management	46
Matching File Names Using Path Name Expansion	49
Practice: Path Name Expansion	52
Lab: Managing Files with Shell Expansion	54
3. Getting Help in Red Hat Enterprise Linux	61
Reading Documentation Using man Command	62
Practice: Using the man Command	65
Reading Documentation Using pinfo Command	67
Practice: Using the pinfo Command	69
Reading Documentation in /usr/share/doc	71
Practice: Viewing Package Documentation	73
Getting Help From Red Hat	74
Practice: Creating and Viewing an SoS Report	79
Lab: Viewing and Printing Help Documentation	81
4. Creating, Viewing, and Editing Text Files	87
Redirecting Output to a File or Program	88
Practice: I/O Redirection and Pipelines	93
Editing Text Files from the Shell Prompt	95
Practice: Editing Files with Vim	98
Editing Text Files with a Graphical Editor	100
Practice: Copying Text Between Windows	103
Lab: Creating, Viewing, and Editing Text Files	105
5. Managing Local Linux Users and Groups	113
Users and Groups	114
Practice: User and Group Concepts	116
Gaining Superuser Access	118
Practice: Running Commands as root	122
Managing Local User Accounts	125

Practice: Creating Users Using Command-line Tools	128
Managing Local Group Accounts	130
Practice: Managing Groups Using Command-line Tools	132
Managing User Passwords	134
Practice: Managing User Password Aging	137
Lab: Managing Local Linux Users and Groups	139
6. Controlling Access to Files with Linux File System Permissions	143
Linux File System Permissions	144
Practice: Interpreting File and Directory Permissions	148
Managing File System Permissions from the Command Line	150
Practice: Managing File Security from the Command Line	153
Managing Default Permissions and File Access	155
Practice: Controlling New File Permissions and Ownership	159
Lab: Controlling Access to Files with Linux File System Permissions	161
7. Monitoring and Managing Linux Processes	165
Processes	166
Practice: Processes	170
Controlling Jobs	172
Practice: Background and Foreground Processes	175
Killing Processes	177
Practice: Killing Processes	181
Monitoring Process Activity	183
Practice: Monitoring Process Activity	187
Lab: Monitoring and Managing Linux Processes	189
8. Controlling Services and Daemons	195
Identifying Automatically Started System Processes	196
Practice: Identify the Status of systemd Units	200
Controlling System Services	202
Practice: Using systemctl to Manage Services	205
Lab: Controlling Services and Daemons	207
9. Configuring and Securing OpenSSH Service	211
Accessing the Remote Command Line with SSH	212
Practice: Accessing the Remote Command Line	215
Configuring SSH Key-based Authentication	217
Practice: Using SSH Key-based Authentication	219
Customizing SSH Service Configuration	220
Practice: Restricting SSH Logins	222
Lab: Configuring and Securing OpenSSH Service	225
10. Analyzing and Storing Logs	229
System Log Architecture	230
Practice: System Logging Components	232
Reviewing Syslog Files	234
Practice: Finding Log Entries	238
Reviewing systemd Journal Entries	239
Practice: Finding Events With journalctl	242
Preserving the systemd Journal	243
Practice: Configure a Persistent systemd Journal	245
Maintaining Accurate Time	246
Practice: Adjusting System Time	250

Lab: Analyzing and Storing Logs	252
11. Managing Red Hat Enterprise Linux Networking	257
Networking Concepts	258
Practice: Networking Concepts	263
Validating Network Configuration	266
Practice: Examining Network Configuration	269
Configuring Networking with nmcli	271
Practice: Configuring Networking with nmcli	276
Editing Network Configuration Files	279
Practice: Editing Network Configuration Files	280
Configuring Host Names and Name Resolution	282
Practice: Configuring Host Names and Name Resolution	285
Lab: Managing Red Hat Enterprise Linux Networking	288
12. Archiving and Copying Files Between Systems	291
Managing Compressed tar Archives	292
Practice: Backing Up and Restoring Files From a tar Archive	297
Copying Files Between Systems Securely	298
Practice: Copying Files Over the Network With scp	300
Synchronizing Files Between Systems Securely	301
Practice: Synchronizing Two Directories Securely With rsync	304
Lab: Archiving and Copying Files Between Systems	306
13. Installing and Updating Software Packages	311
Attaching Systems to Subscriptions for Software Updates	312
Practice: Red Hat Subscription Management	318
RPM Software Packages and Yum	320
Practice: RPM Software Packages	323
Managing Software Updates with yum	325
Practice: Installing and Updating Software with yum	332
Enabling yum Software Repositories	336
Practice: Enabling Software Repositories	339
Examining RPM Package Files	341
Practice: Working with RPM Package Files	345
Lab: Installing and Updating Software Packages	347
14. Accessing Linux File Systems	351
Identifying File Systems and Devices	352
Practice: Identifying File Systems and Devices	355
Mounting and Unmounting File Systems	357
Practice: Mounting and Unmounting File Systems	360
Making Links Between Files	362
Practice: Making Links Between Files	364
Locating Files on the System	365
Practice: Locating Files on the System	372
Lab: Accessing Linux File Systems	374
15. Using Virtualized Systems	379
Managing a Local Virtualization Host	380
Practice: Managing a Local Virtualization Host	386
Installing a New Virtual Machine	388
Practice: Installing a New Virtual Machine	397
Chapter Test: Using Virtualized Systems	399

16. Comprehensive Review	403
Red Hat System Administration I Comprehensive Review	404
Lab: Comprehensive Review	408

Document Conventions

Notes and Warnings



Note

"Notes" are tips, shortcuts or alternative approaches to the task at hand. Ignoring a note should have no negative consequences, but you might miss out on a trick that makes your life easier.



Important

"Important" boxes detail things that are easily missed: configuration changes that only apply to the current session, or services that need restarting before an update will apply. Ignoring a box labeled "Important" will not cause data loss, but may cause irritation and frustration.



Warning

"Warnings" should not be ignored. Ignoring warnings will most likely cause data loss.



References

"References" describe where to find external documentation relevant to a subject.

Introduction

Red Hat System Administration I

Red Hat System Administration I (RH124) is designed for IT professionals without previous Linux system administration experience. The course is intended to provide students with Linux administration "survival skills" by focusing on core administration tasks. *Red Hat System Administration I* also provides a foundation for students planning to become full-time Linux system administrators by introducing key command-line concepts and enterprise-level tools. These concepts are further developed in the follow-on course, *Red Hat System Administration II* (RH134).

Course objectives

- Gain sufficient skill to perform core system administration tasks on Red Hat Enterprise Linux.
- Build foundational skills needed by an RHCSA-certified Red Hat Enterprise Linux system administrator.

Audience

- IT professionals across a broad range of disciplines who need to perform essential Linux administration tasks, including installation, establishing network connectivity, managing physical storage and basic security administration.

Prerequisites

- There are no formal prerequisites for this course; however, previous system administration experience on other operating systems will be very beneficial.

Orientation to the Classroom Environment

In this course, students will do most hands-on practice exercises and lab work with two computer systems, which will be referred to as **desktop** and **server**. These machines have the host names desktopX.example.com and serverX.example.com, where the X in the computers' host names will be a number that will vary from student to student. Both machines have a standard user account, *student*, with the password *student*. The **root** password on both systems is *redhat*.

In a live instructor-led classroom, students will be assigned a physical computer ("foundationX"), which will be used to access these two machines. The **desktop** and **server** systems are virtual machines running on that host. Students should log into this machine as user *kiosk* with the password *redhat*.

On foundationX, a special command called **rht-vmctl** is used to work with the **desktop** and **server** machines. The commands in the following table should be run as the *kiosk* user on foundationX, and can be used with **server** (as in the examples) or **desktop**.

rht-vmctl Commands

Action	Command
Start server machine	rht-vmctl start server
View "physical console" to log in and work with server machine	rht-vmctl view server
Reset server machine to its previous state and restart virtual machine	rht-vmctl reset server

At the start of a lab exercise, if the instruction "reset your server" appears, that means the command **rht-vmctl reset server** should be run in a prompt on the foundationX system as user *kiosk*. Likewise, if the instruction "reset your desktop" appears, that means the command **rht-vmctl reset desktop** should be run on foundationX as user *kiosk*.

Each student is on the IPv4 network 172.25.X.0/24, where the X matches the number of their desktopX and serverX systems. The instructor runs a central utility server, classroom.example.com, which acts as a router for the classroom networks and provides DNS, DHCP, HTTP, and other content services.

Classroom Machines

Machine name	IP addresses	Role
desktopX.example.com	172.25.X.10	Student "client" computer
serverX.example.com	172.25.X.11	Student "server" computer
classroom.example.com	172.25.254.254	Classroom utility server

Internationalization

Language support

Red Hat Enterprise Linux 7 officially supports 22 languages: English, Assamese, Bengali, Chinese (Simplified), Chinese (Traditional), French, German, Gujarati, Hindi, Italian, Japanese, Kannada, Korean, Malayalam, Marathi, Odia, Portuguese (Brazilian), Punjabi, Russian, Spanish, Tamil, and Telugu.

Per-user language selection

Users may prefer to use a different language for their desktop environment than the system-wide default. They may also want to set their account to use a different keyboard layout or input method.

Language settings

In the GNOME desktop environment, the user may be prompted to set their preferred language and input method on first login. If not, then the easiest way for an individual user to adjust their preferred language and input method settings is to use the **Region & Language** application. Run the command **gnome-control-center region**, or from the top bar, select (User) > Settings. In the window that opens, select **Region & Language**. The user can click the **Language** box and select their preferred language from the list that appears. This will also update the **Formats** setting to the default for that language. The next time the user logs in, these changes will take full effect.

These settings affect the GNOME desktop environment and any applications, including **gnome-terminal**, started inside it. However, they do not apply to that account if accessed through an **ssh** login from a remote system or a local text console (such as **tty2**).



Note

A user can make their shell environment use the same **LANG** setting as their graphical environment, even when they log in through a text console or over **ssh**. One way to do this is to place code similar to the following in the user's **~/.bashrc** file. This example code will set the language used on a text login to match the one currently set for the user's GNOME desktop environment:

```
i=$(grep 'Language=' /var/lib/AccountService/users/${USER} \
    | sed 's/Language=//')
if [ "$i" != "" ]; then
    export LANG=$i
fi
```

Japanese, Korean, Chinese, or other languages with a non-Latin character set may not display properly on local text consoles.

Individual commands can be made to use another language by setting the **LANG** variable on the command line:

```
[user@host ~]$ LANG=fr_FR.utf8 date
```

jeu. avril 24 17:55:01 CDT 2014

Subsequent commands will revert to using the system's default language for output. The **locale** command can be used to check the current value of **LANG** and other related environment variables.

Input method settings

GNOME 3 in Red Hat Enterprise Linux 7 automatically uses the IBus input method selection system, which makes it easy to change keyboard layouts and input methods quickly.

The **Region & Language** application can also be used to enable alternative input methods. In the **Region & Language** application's window, the **Input Sources** box shows what input methods are currently available. By default, **English (US)** may be the only available method. Highlight **English (US)** and click the **Keyboard** icon to see the current keyboard layout.

To add another input method, click the **+** button at the bottom left of the **Input Sources** window. An **Add an Input Source** window will open. Select your language, and then your preferred input method or keyboard layout.

Once more than one input method is configured, the user can switch between them quickly by typing **Super+Space** (sometimes called **Windows+Space**). A *status indicator* will also appear in the GNOME top bar, which has two functions: It indicates which input method is active, and acts as a menu that can be used to switch between input methods or select advanced features of more complex input methods.

Some of the methods are marked with gears, which indicate that those methods have advanced configuration options and capabilities. For example, the Japanese **Japanese (Kana Kanji)** input method allows the user to pre-edit text in Latin and use **Down Arrow** and **Up Arrow** keys to select the correct characters to use.

US English speakers may find also this useful. For example, under **English (United States)** is the keyboard layout **English (international AltGr dead keys)**, which treats **AltGr** (or the right **Alt**) on a PC 104/105-key keyboard as a "secondary-shift" modifier key and dead key activation key for typing additional characters. There are also Dvorak and other alternative layouts available.



Note

Any Unicode character can be entered in the GNOME desktop environment if the user knows the character's Unicode code point, by typing **Ctrl+Shift+U**, followed by the code point. After **Ctrl+Shift+U** has been typed, an underlined **u** will be displayed to indicate that the system is waiting for Unicode code point entry.

For example, the lowercase Greek letter lambda has the code point U+03BB, and can be entered by typing **Ctrl+Shift+U**, then **03bb**, then **Enter**.

System-wide default language settings

The system's default language is set to US English, using the UTF-8 encoding of Unicode as its character set (**en_us.utf8**), but this can be changed during or after installation.

From the command line, **root** can change the system-wide locale settings with the **localectl** command. If **localectl** is run with no arguments, it will display the current system-wide locale settings.

To set the system-wide language, run the command **localectl set-locale LANG=locale**, where *locale* is the appropriate **\$LANG** from the "Language Codes Reference" table in this chapter. The change will take effect for users on their next login, and is stored in **/etc/locale.conf**.

```
[root@host ~]# localectl set-locale LANG=fr_FR.utf8
```

In GNOME, an administrative user can change this setting from **Region & Language** and clicking the **Login Screen** button at the upper-right corner of the window. Changing the **Language** of the login screen will also adjust the system-wide default language setting stored in the **/etc/locale.conf** configuration file.



Important

Local text consoles such as **tty2** are more limited in the fonts that they can display than **gnome-terminal** and **ssh** sessions. For example, Japanese, Korean, and Chinese characters may not display as expected on a local text console. For this reason, it may make sense to use English or another language with a Latin character set for the system's text console.

Likewise, local text consoles are more limited in the input methods they support, and this is managed separately from the graphical desktop environment. The available global input settings can be configured through **localectl** for both local text virtual consoles and the X11 graphical environment. See the **localectl(1)**, **kbd(4)**, and **vconsole.conf(5)** man pages for more information.

Language packs

When using non-English languages, you may want to install additional "language packs" to provide additional translations, dictionaries, and so forth. To view the list of available langpacks, run **yum langavailable**. To view the list of langpacks currently installed on the system, run **yum langlist**. To add an additional langpack to the system, run **yum langinstall code**, where *code* is the code in square brackets after the language name in the output of **yum langavailable**.



References

locale(7), **localectl(1)**, **kbd(4)**, **locale.conf(5)**, **vconsole.conf(5)**,
unicode(7), **utf-8(7)**, and **yum-langpacks(8)** man pages

Conversions between the names of the graphical desktop environment's X11 layouts and their names in **localectl** can be found in the file **/usr/share/X11/xkb/rules/base.lst**.

Language Codes Reference

Language Codes

Language	\$LANG value
English (US)	en_US.utf8
Assamese	as_IN.utf8
Bengali	bn_IN.utf8
Chinese (Simplified)	zh_CN.utf8
Chinese (Traditional)	zh_TW.utf8
French	fr_FR.utf8
German	de_DE.utf8
Gujarati	gu_IN.utf8
Hindi	hi_IN.utf8
Italian	it_IT.utf8
Japanese	ja_JP.utf8
Kannada	kn_IN.utf8
Korean	ko_KR.utf8
Malayalam	ml_IN.utf8
Marathi	mr_IN.utf8
Odia	or_IN.utf8
Portuguese (Brazilian)	pt_BR.utf8
Punjabi	pa_IN.utf8
Russian	ru_RU.utf8
Spanish	es_ES.utf8
Tamil	ta_IN.utf8
Telugu	te_IN.utf8



CHAPTER 1

ACCESSING THE COMMAND LINE

Overview	
Goal	To log into a Linux system and run simple commands using the shell.
Objectives	<ul style="list-style-type: none">• Use Bash shell syntax to enter commands at a Linux console.• Launch applications in a GNOME desktop environment.• Use Bash features to run commands from a shell prompt using fewer keystrokes.
Sections	<ul style="list-style-type: none">• Accessing the Command Line Using the Local Console (and Practice)• Accessing the Command Line Using the Desktop (and Practice)• Executing Commands Using the Bash Shell (and Practice)
Lab	<ul style="list-style-type: none">• Accessing the Command Line

Accessing the Command Line Using the Local Console

Objectives

After completing this section, students should be able to log into a Linux system on a local text console and run simple commands using the shell.

The bash shell

A *command line* is a text-based interface which can be used to input instructions to a computer system. The Linux command line is provided by a program called the *shell*. Over the long history of UNIX-like systems, many shells have been developed. The default shell for users in Red Hat Enterprise Linux is the **GNU Bourne-Again Shell (bash)**. **Bash** is an improved version of one of the most successful shells used on UNIX-like systems, the **Bourne Shell (sh)**.

When a shell is used interactively, it displays a string when it is waiting for a command from the user. This is called the *shell prompt*. When a regular user starts a shell, the default prompt ends with a \$ character.

```
[student@desktopX ~]$
```

The \$ is replaced by a # if the shell is running as the superuser, **root**. This makes it more obvious that it is a superuser shell, which helps to avoid accidents and mistakes in the privileged account.

```
[root@desktopX ~]#
```

Using **bash** to execute commands can be powerful. The **bash** shell provides a scripting language that can support automation of tasks. The shell has additional capabilities that can simplify or make possible operations that are hard to accomplish efficiently with graphical tools.



Note

The **bash** shell is similar in concept to the command line interpreter found in recent versions of Microsoft Windows **cmd.exe**, although **bash** has a more sophisticated scripting language. It is also similar to Windows PowerShell in Windows 7 and Windows Server 2008 R2. Mac OS X administrators who use the Macintosh's **Terminal** utility may be pleased to note that **bash** is the default shell in Mac OS X.

Virtual consoles

Users access the **bash** shell through a *terminal*. A terminal provides a keyboard for user input and a display for output. On text-based installations, this can be the Linux machine's *physical console*, the hardware keyboard and display. Terminal access can also be configured through serial ports.

Another way to access a shell is from a *virtual console*. A Linux machine's physical console supports multiple virtual consoles which act like separate terminals. Each virtual console supports an independent login session.

If the graphical environment is available, it will run on the *first* virtual console in Red Hat Enterprise Linux 7. Five additional text login prompts are available on consoles two through six (or one through five if the graphical environment is turned off). With a graphical environment running, access a text login prompt on a virtual console by holding **Ctrl+Alt** and pressing a function key (**F2** through **F6**). Press **Ctrl+Alt+F1** to return to the first virtual console and the graphical desktop.



Important

In the pre-configured virtual images delivered by Red Hat, login prompts have been disabled in the virtual consoles.



Note

In Red Hat Enterprise Linux 5 and earlier, the first *six* virtual consoles always provided text login prompts. When the graphical environment was launched, it ran on virtual console seven (accessed through **Ctrl+Alt+F7**).

Shell basics

Commands entered at the shell prompt have three basic parts:

- *Command* to run
- *Options* to adjust the behavior of the command
- *Arguments*, which are typically targets of the command

The *command* is the name of the program to run. It may be followed by one or more *options*, which adjust the behavior of the command or what it will do. Options normally start with one or two dashes (-**a** or --**all**, for example) to distinguish them from arguments. Commands may also be followed by one or more *arguments*, which often indicate a target that the command should operate on.

For example, the command line **usermod -L morgan** has a command (**usermod**), an option (-**L**), and an argument (**morgan**). The effect of this command is to lock the password on user morgan's account.

To use a command effectively, a user needs to know what options and arguments it takes and in what order it expects them (the *syntax* of the command). Most commands have a --**help** option. This causes the command to print a description of what it does, a "usage statement" that describes the command's syntax, and a list of the options it accepts and what they do.

Usage statements may seem complicated and difficult to read. They become much simpler to understand once a user becomes familiar with a few basic conventions:

- Square brackets, [], surround optional items.
- Anything followed by ... represents an arbitrary-length list of items of that type.
- Multiple items separated by pipes, | , means only one of them can be specified.

- Text in angle brackets, <>, represents variable data. For example, <filename> means "insert the filename you wish to use here". Sometimes these variables are simply written in capital letters (e.g., FILENAME).

Consider the first usage statement for the **date** command:

```
[student@desktopX ~]$ date --help  
date [OPTION]... [+FORMAT]
```

This indicates that **date** can take an optional list of options ([OPTION]...), followed by an optional format string, prefixed with a plus character, +, that defines how the current date should be displayed ([+FORMAT]). Since both of these are optional, **date** will work even if it is not given options or arguments (it will print the current date and time using its default format).



Note

The **man** page for a command has a SYNOPSIS section that provides information about the command's syntax. The **man-pages(7)** man page describes how to interpret all the square brackets, vertical bars, and so forth that users see in SYNOPSIS or a usage message.

When a user is finished using the shell and wants to quit, there are a couple of ways to end the session. The **exit** command terminates the current shell session. Another way to finish a session is by typing **Ctrl+D**.



References

intro(1), **bash(1)**, **console(4)**, **pts(4)**, and **man-pages(7)** man pages

Note: Some details of the console(4) man page, involving init(8) and inittab(5), are outdated.

Practice: Local Console Access Terms

Quiz

Match the following items to their counterparts in the table.

Argument	Command	Option	Physical console
Prompt	Shell	Terminal	Virtual console

Description	Term
The interpreter that executes commands typed as strings.	
The visual cue that indicates an interactive shell is waiting for the user to type a command.	
The name of a program to run.	
The part of the command line that adjusts the behavior of a command.	
The part of the command line that specifies the target that the command should operate on.	
The hardware display and keyboard used to interact with a system.	
One of multiple logical consoles that can each support an independent login session.	

Description	Term
An interface that provides a display for output and a keyboard for input to a shell session.	

Solution

Match the following items to their counterparts in the table.

Description	Term
The interpreter that executes commands typed as strings.	Shell
The visual cue that indicates an interactive shell is waiting for the user to type a command.	Prompt
The name of a program to run.	Command
The part of the command line that adjusts the behavior of a command.	Option
The part of the command line that specifies the target that the command should operate on.	Argument
The hardware display and keyboard used to interact with a system.	Physical console
One of multiple logical consoles that can each support an independent login session.	Virtual console
An interface that provides a display for output and a keyboard for input to a shell session.	Terminal

Accessing the Command Line Using the Desktop

Objectives

After completing this section, students should be able to log into the Linux system using the GNOME 3 desktop environment to run commands from a shell prompt in a terminal program.

The GNOME desktop environment

The *desktop environment* is the graphical user interface on a Linux system. The default desktop environment in Red Hat Enterprise Linux 7 is provided by **GNOME 3**. It provides an integrated desktop for users and a unified development platform on top of a graphical framework provided by the **X Window System**.

The **GNOME Shell** provides the core user interface functions for the GNOME desktop environment. The **gnome-shell** application is highly customizable. By default, RHEL 7 users use the "GNOME Classic" theme for **gnome-shell**, which is similar to the GNOME 2 desktop environment. Another available option is the "modern" GNOME 3 theme used by the upstream GNOME project. Either theme can be selected persistently at login by selecting the gear icon next to the **Sign In** button when entering the user's password.

The first time a new user logs in, an initial setup program runs to help them configure basic account settings. The **GNOME Help** application is then started on the **Getting Started with GNOME** screen. This screen includes videos and documentation to help orient new users to the GNOME 3 environment. **GNOME Help** can be quickly started by typing **F1** in **gnome-shell**, by selecting **Applications > Documentation > Help**, or by running the **yelp** command.



Figure 1.1: An empty GNOME 3 desktop

Parts of the GNOME Shell

The various parts of the GNOME Shell have specific names and purposes. These parts include the following:

- **top bar:** The bar that runs along the top of the screen. The top bar provides the **Applications** and **Places** menus, and controls for volume, networking, calendar access, and selecting between keyboard input methods (if there are more than one configured). Under the menu for the user's name are options to adjust account settings, lock the screen, switch users, log out of the system, or shut it down.
- **Applications menu:** This menu on the top bar provides a way to start applications, categorized into submenus. The **Activities Overview** can also be started from this menu.
- **Places menu:** This menu to the right of the **Applications** menu provides quick access through a graphical file manager to important menus in the user's home directory, to **/**, and to exports and file shares on the network.
- **window list:** The bar that runs along the bottom of the screen. The window list provides an easy way to access, minimize, and restore all windows in the current workspace. On the right corner is an indicator to tell the user which workspace they are on and how many are available.
- **message tray:** The message tray provides a way to review notifications sent by applications or system components to GNOME. If a notification occurs, normally the notification first appears briefly as a single line at the bottom of the screen, and a persistent indicator appears in the lower right corner to inform the user of how many notifications have been recently received.

The message tray can be opened to review these notifications by clicking the indicator or typing **Super+M**. The message tray can be closed by typing either **Esc** or **Super+M** again.

- **Activities Overview:** This is a special mode that helps a user organize windows and start applications. The Activities Overview can be started by selecting **Applications > Activities Overview**. A faster way is to press the **Super** key (sometimes called the **Windows** key), found near the lower left corner of an IBM PC 104/105-key keyboard. The three main areas of the Activities Overview are the **dash** on the left side of the screen, the **windows overview** in the center of the screen, and the **workspace selector** on the right side of the screen. The Activities Overview can be exited by pressing the **Esc** key.
- **dash:** This is a configurable list of icons of the user's favorite applications, applications which are currently running, and a **grid** button which can be used to select arbitrary applications. Applications can be started by clicking on one of the icons or by using the grid button to find a less commonly used application. The dash is also sometimes called the **dock**.

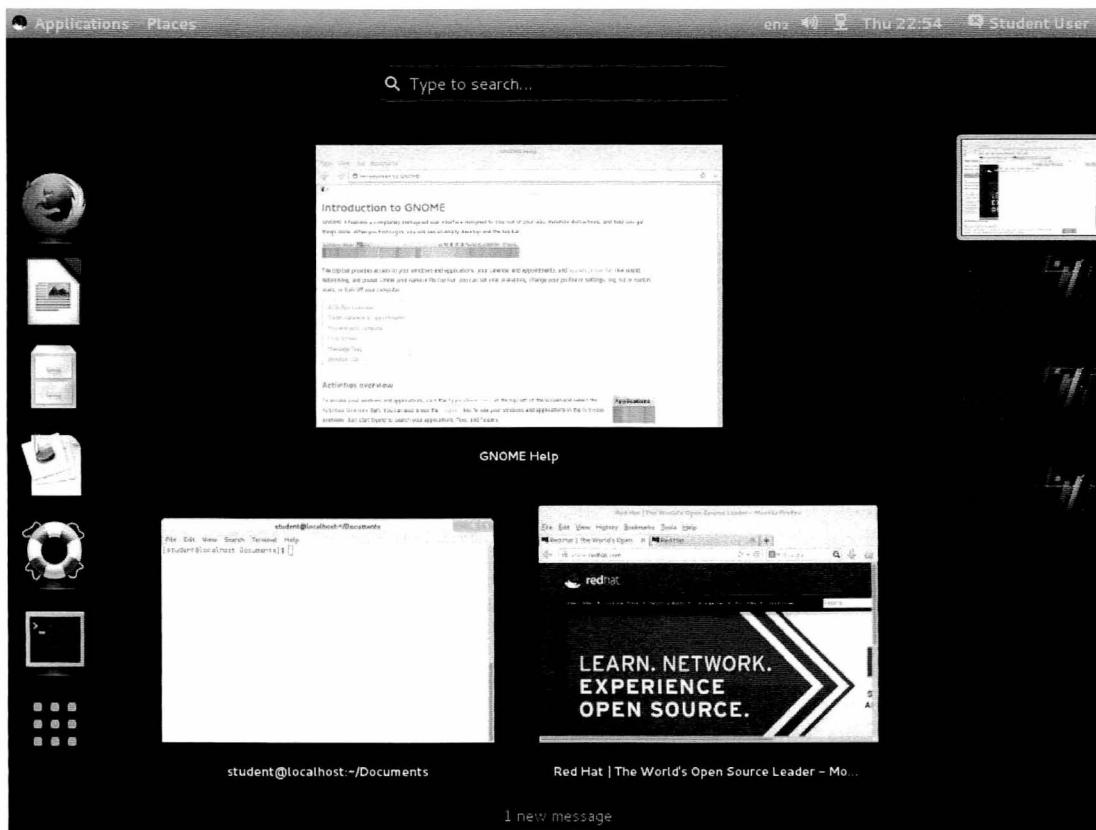


Figure 1.2: The GNOME 3 Activities Overview

Workspaces

Workspaces are separate desktop screens which have different application windows. These can be used to organize the working environment by grouping open application windows by task. For example, windows being used to perform a particular system maintenance activity (such as setting up a new remote server) can be grouped in one workspace, while email and other communication applications can be grouped in another workspace.

There are three methods for switching between workspaces. One method is to click the indicator in the right corner of the window list and select the desired workspace. Another, perhaps the fastest, is to type **Ctrl+Alt+UpArrow** or **Ctrl+Alt+DownArrow** to switch between workspaces sequentially. A third is to switch to the **Activities Overview** and click the desired workspace.

An advantage of using the **Activities Overview** is that windows can be clicked and dragged between the current workspace and one of the others by using the **workspace selector** on the right side of the screen and the **windows overview** in the center of the screen.

Starting a terminal

To get a shell prompt in GNOME, start a graphical terminal application such as **GNOME Terminal**. There are several ways to do this. Here are the three most commonly used methods:

- Select **Applications > Utilities > Terminal**.
- On an empty desktop, right-click, or press the **Menu** key, and select **Open in Terminal** from the context menu that appears.
- From the **Activities Overview**, select **Terminal** from the **dash** (either from the favorites area or by finding it with either the **grid** button (inside Utilities grouping) or the search field at the top of the **windows overview**).

When a terminal window is opened, a shell prompt displays for the user that started the graphical terminal program. The shell prompt and the terminal window's title bar will indicate the current user name, host name, and working directory.

Locking the screen or logging out

Locking the screen, or logging out entirely, can be done from the menu for the user's name on the far right side of the top bar.

To lock the screen, select **(User) > Lock** or type **Ctrl+Alt+L**. The screen will lock if the graphical session is idle for a few minutes.

A **lock screen curtain** will appear that shows the system time and the name of the logged-in user. To unlock the screen, press **Enter** or **Space** to raise the lock screen curtain, then enter the user's password on the **lock screen**.

To log out and end the current graphical login session, select **(User) > Log Out** from the top bar. A dialog window will appear, giving the option to **Cancel** the log out within 60 seconds, or confirm the **Log Out** action.

Powering off or rebooting the system

To shut down the system, select **(User) > Power Off** from the top bar or type **Ctrl+Alt+Del**. In the dialog that appears, the user can choose to **Power Off**, **Restart** the machine, or **Cancel** the operation. If the user does not make a choice in this dialog, the system will automatically shut down after 60 seconds.



References

GNOME Help

- **yelp**

GNOME Help: *Getting Started with GNOME*

- **yelp help:gnome-help/getting-started**

Practice: The GNOME 3 Desktop Environment

Guided exercise

In this lab, you will log in through the graphical display manager as a regular user to become familiar with the GNOME Classic desktop environment provided by GNOME 3.

Outcome:

A basic orientation to the GNOME 3 desktop environment.

Before you begin...

Access the graphical login screen of **desktopX.example.com**.



Important

There are two virtual machines available for lab exercises, a desktop machine (generically called **desktopX**) and a server (generically called **serverX**).

Take care to keep straight which virtual machine an exercise wants you to use.

Do each of the following tasks on the **desktopX** machine. Mark each task as it is completed.

- 1. Log in as **student** using the password **student**.
 - 1.1. At the GNOME login screen, click the **student** user account. Enter **student** when prompted for the password.
 - 1.2. Click **Sign In** once the password has been typed in.
- 2. Change the password for **student** from **password** to **55TurnK3y**.
 - 2.1. The simplest approach is to open **GNOME Terminal** and use the **passwd** command at the shell prompt.

On the empty desktop, press the **Menu** key or right-click with the mouse to open the context menu.
 - 2.2. Select **Open in Terminal**.
 - 2.3. In the terminal window that appears, type **passwd** at the shell prompt. Follow the instructions provided by the program to change the **student** password from **student** to **55TurnK3y**.
- 3. Log out.
 - 3.1. Select the **student > Log Out** menu item.
 - 3.2. Click the **Log Out** button in the confirmation window that appears.
- 4. Log back in as **student** with the new password of **55TurnK3y**.
 - 4.1. At the GNOME login screen, click the **student** user account. Enter **55TurnK3y** when prompted for the password.

- 4.2. Click **Sign In** once the password has been typed in.
- 5. Lock the screen.
 - 5.1. Select the **student > Lock** menu item.
- 6. Unlock the screen.
 - 6.1. Press **Enter** to lift the lock screen curtain.
 - 6.2. In the **Password** field, enter **55TurnK3y** as the password.
 - 6.3. Click the **Unlock** button. The GNOME desktop should reappear.
- 7. Determine how to shut down **desktopX** from the graphical interface, but **Cancel** the operation without shutting down the system.
 - 7.1. Select the **student > Power Off** menu item.
 - 7.2. Click the **Cancel** button in the confirmation screen that appears.

Executing Commands Using the Bash Shell

Objectives

After completing this section, students should be able to save time running commands from a shell prompt using Bash shortcuts.

Basic command syntax

The **GNU Bourne-Again Shell (bash)** is a program that interprets commands typed in by the user. Each string typed into the shell can have up to three parts: the command, options (that begin with a - or --), and arguments. Each word typed into the shell is separated from each other with spaces. Commands are the names of programs that are installed on the system. Each command has its own options and arguments.

The **Enter** key is pressed when a user is ready to execute a command. Each command is typed on a separate line and the output from each command displays before the shell displays a prompt. If a user wants to type more than one command on a single line, a semicolon, ;, can be used as a command separator. A semicolon is a member of a class of characters called *metacharacters* that has special meanings for **bash**.

Examples of simple commands

The **date** command is used to display the current date and time. It can also be used by the superuser to set the system clock. An argument that begins with a plus sign (+) specifies a format string for the date command.

```
[student@desktopX ~]$ date
Sat Apr  5 08:13:50 PDT 2014
[student@desktopX ~]$ date +%R
08:13
[student@desktopX ~]$ date +%x
04/05/2014
```

The **passwd** command changes a user's own password. The original password for the account must be specified before a change will be allowed. By default, **passwd** is configured to require a strong password, consisting of lowercase letters, uppercase letters, numbers, and symbols, and is not based on a dictionary word. The superuser can use the **passwd** command to change other users' passwords.

```
[student@desktopX ~]$ passwd
Changing password for user student.
Changing password for student.
(current) UNIX password: old_password
New password: new_password
Retype new password: new_password
passwd: all authentication tokens updated successfully.
```

Linux does not require file name extensions to classify files by type. The **file** command scans the beginning of a file's contents and displays what type it is. The files to be classified are passed as arguments to the command.

```
[student@desktopX ~]$ file /etc/passwd
```

```
/etc/passwd: ASCII text
[student@desktopX ~]$ file /bin/passwd
/bin/passwd: setuid ELF 64-bit LSB shared object, x86-64, version 1
(SYSV), dynamically linked (uses shared libs), for GNU/Linux 2.6.32,
BuildID[sha1]=0x91a7160a019b7f5f754264d920e257522c5bce67, stripped
[student@desktopX ~]$ file /home
/home: directory
```

The **head** and **tail** commands display the beginning and end of a file respectively. By default, these commands display 10 lines, but they both have a **-n** option that allows a different number of lines to be specified. The file to display is passed as an argument to these commands.

```
[student@desktopX ~]$ head /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
operator:x:11:0:operator:/root:/sbin/nologin
[student@desktopX ~]$ tail -n 3 /etc/passwd
gdm:x:42:42::/var/lib/gdm:/sbin/nologin
gnome-initial-setup:x:993:991::/run/gnome-initial-setup/:/sbin/nologin
tcpdump:x:72:72:::/sbin/nologin
```

The **wc** command counts lines, words, and characters in a file. It can take a **-l**, **-w**, or **-c** option to display only the lines, words, or characters, respectively.

```
[student@desktopX ~]$ wc /etc/passwd
39    70 2005 /etc/passwd
[student@desktopX ~]$ wc -l /etc/passwd
39 /etc/passwd
[student@desktopX ~]$ wc -c /etc/group /etc/hosts
843 /etc/group
227 /etc/hosts
1070 total
```

Tab completion

Tab completion allows a user to quickly complete commands or file names once they have typed enough at the prompt to make it unique. If the characters typed are not unique, pressing the **Tab** key twice displays all commands that begin with the characters already typed.

```
[student@desktopX ~]$ pas<Tab><Tab>
passwd      paste      pasuspender
[student@desktopX ~]$ pass<Tab>
[student@desktopX ~]$ passwd
Changing password for user student.
Changing password for student.
(current) UNIX password:
```

Tab completion can be used to complete file names when typing them as arguments to commands. When **Tab** is pressed, it will complete the file name as much as it can. Pressing **Tab** a second time causes the shell to list all of the files that are matched by the current pattern.

Type additional characters until the name is unique, then use tab completion to finish off the command line.

```
[student@desktopX ~]$ ls /etc/pas<Tab>
[student@desktopX ~]$ ls /etc/passwd<Tab>
passwd  passwd-
```

Arguments and options can be matched with tab completion for many commands. The **useradd** command is used by the superuser, **root**, to create additional users on the system. It has many options that can be used to control how that command behaves. Tab completion following a partial option can be used to complete the option without a lot of typing.

```
[root@desktopX ~]# useradd --<Tab><Tab>
--base-dir      --groups          --no-log-init    --shell
--comment       --help            --non-unique     --skel
--create-home   --home-dir       --no-user-group  --system
--defaults      --inactive       --password      --uid
--expiredate   --key            --root          --user-group
--gid           --no-create-home --selinux-user
[root@desktopX ~]# useradd --
```

Command history

The **history** command displays a list of previously executed commands prefixed with a command number.

The exclamation point character, **!**, is a metacharacter that is used to expand previous commands without having to retype them. **!number** expands to the command matching the number specified. **!string** expands to the most recent command that begins with the string specified.

```
[student@desktopX ~]$ history
...Output omitted...
23  clear
24  who
25  pwd
26  ls /etc
27  uptime
28  ls -l
29  date
30  history
[student@desktopX ~]$ !ls
ls -l
total 0
drwxr-xr-x. 2 student student 6 Mar 29 21:16 Desktop
...Output omitted...
[student@desktopX ~]$ !26
ls /etc
abrt                  hosts             pulse
adjtime               hosts.allow       purple
aliases               hosts.deny        qemu-ga
...Output omitted...
```

The arrow keys can be used to navigate through previous command lines in the shell's history. **Up Arrow** edits the previous command in the history list. **Down Arrow** edits the next command in the history list. Use this key when the **Up Arrow** has been pressed too many times. **Left Arrow** and **Right Arrow** move the cursor left and right in the current command line being edited.

The **Esc+.** key combination causes the shell to copy the last word of the previous command on the current command line where the cursor is. If used repeatedly, it will continue to go through earlier commands.

Editing the command line

When used interactively, **bash** has a command line-editing feature. This allows the user to use text editor commands to move around within and modify the current command being typed. Using the arrow keys to move within the current command and to step through the command history was introduced earlier in this session. More powerful editing commands are introduced in the following table.

Useful command line-editing shortcuts

Shortcut	Description
Ctrl+a	Jump to the beginning of the command line.
Ctrl+e	Jump to the end of the command line.
Ctrl+u	Clear from the cursor to the beginning of the command line.
Ctrl+k	Clear from the cursor to the end of the command line.
Ctrl+Left Arrow	Jump to the beginning of the previous word on the command line.
Ctrl+Right Arrow	Jump to the beginning of the next word on the command line.
Ctrl+r	Search the history list of commands for a pattern.

There are several other command line-editing commands available, but these are the most useful commands for beginning users. The other commands can be found in the **bash(1)** man page.

R

References

bash(1), date(1), file(1), head(1), passwd(1), tail(1), and wc(1) man pages

Practice: Bash Commands and Keyboard Shortcuts

Quiz

Match the following Bash shortcuts to their descriptions in the table.

<i>!number</i>	<i>!string</i>	<i>;</i>	<i>Ctrl+Left Arrow</i>	<i>Ctrl+a</i>
<i>Ctrl+k</i>	<i>Esc+. </i>	<i>Tab</i>	<i>history</i>	

Description	Shell command
Jump to the beginning of the previous word on the command line.	
Separate commands on the same line.	
Clear from the cursor to the end of the command line.	
Re-execute a recent command by matching the command name.	
Shortcut used to complete commands, file names, and options.	
Re-execute a specific command in the history list.	
Jump to the beginning for the command line.	

Description	Shell command
Display the list of previous commands.	
Copy the last argument of previous commands.	

Solution

Match the following Bash shortcuts to their descriptions in the table.

Description	Shell command
Jump to the beginning of the previous word on the command line.	Ctrl+Left Arrow
Separate commands on the same line.	;
Clear from the cursor to the end of the command line.	Ctrl+k
Re-execute a recent command by matching the command name.	!string
Shortcut used to complete commands, file names, and options.	Tab
Re-execute a specific command in the history list.	!number
Jump to the beginning for the command line.	Ctrl+a
Display the list of previous commands.	history
Copy the last argument of previous commands.	Esc+.

Lab: Accessing the Command Line

Performance checklist

In this lab, you will use the Bash shell to efficiently execute commands using shell metacharacters.

Resources:	
Files:	/usr/bin/clean-binary-files

Outcomes:

- Practice using shell command line editing and history functions to efficiently execute commands with minor changes.
- Change the password of the **student** user to **T3st1ngT1me**.
- Execute commands used to identify file types and display parts of text files.

Before you begin...

Perform the following steps on desktopX.

1. Log into your **desktopX** system's graphical login screen as **student**.
2. Open a terminal window that will provide a **bash** prompt.
3. Change **student**'s password to **T3st1ngT1me**.
4. Display the current time and date.
5. Display the current time in the following format: HH:MM:SS A/PM. Hint: The format string that displays that output is **%r**.
6. What kind of file is **/usr/bin/clean-binary-files**? Is it readable by humans?
7. Use the **wc** command and **bash** shortcuts to display the size of **/usr/bin/clean-binary-files**.
8. Display the first 10 lines of **/usr/bin/clean-binary-files**.
9. Display the last 10 lines at the bottom of the **/usr/bin/clean-binary-files** file.
10. Repeat the previous command, but use the **-n 20** option to display the last 20 lines in the file. Use command line editing to accomplish this with a minimal amount of keystrokes.
11. Execute the **date** command without any arguments to display the current date and time.
12. Use **bash** history to display just the time.
13. Finish your session with the **bash** shell.

Solution

In this lab, you will use the Bash shell to efficiently execute commands using shell metacharacters.

Resources:	
Files:	/usr/bin/clean-binary-files

Outcomes:

- Practice using shell command line editing and history functions to efficiently execute commands with minor changes.
- Change the password of the **student** user to **T3st1ngT1me**.
- Execute commands used to identify file types and display parts of text files.

Before you begin...

Perform the following steps on desktopX.

1. Log into your **desktopX** system's graphical login screen as **student**.
2. Open a terminal window that will provide a **bash** prompt.
Select Applications > Utilities > Terminal.
3. Change **student**'s password to **T3st1ngT1me**.

Use the **passwd** command to change the password. Be sure to provide the original password, **student**, first.

```
[student@desktopX ~]$ passwd
Changing password for user student.
Changing password for student.
(current) UNIX password: student
New password: T3st1ngT1me
Retype new password: T3st1ngT1me
passwd: all authentication tokens updated successfully.
```

4. Display the current time and date.

```
[student@desktopX ~]$ date
Thu Apr  3 10:13:04 PDT 2014
```

5. Display the current time in the following format: HH:MM:SS A/PM. Hint: The format string that displays that output is **%r**.

Specify the **+%r** argument to **date**.

```
[student@desktopX ~]$ date +%r
10:14:07 AM
```

6. What kind of file is **/usr/bin/clean-binary-files**? Is it readable by humans?

Use the **file** command to determine its file type.

```
[student@desktopX ~]$ file /usr/bin/clean-binary-files  
/usr/bin/clean-binary-files: POSIX shell script, ASCII text executable
```

7. Use the **wc** command and **bash** shortcuts to display the size of **/usr/bin/clean-binary-files**.

The easiest shortcut to use is **Esc+.** to reuse the argument from the previous command.

```
[student@desktopX ~]$ wc <Esc>.  
[student@desktopX ~]$ wc /usr/bin/clean-binary-files  
594 1780 13220 /usr/bin/clean-binary-files
```

8. Display the first 10 lines of **/usr/bin/clean-binary-files**.

The **head** command displays the beginning of the file. Did you use the **bash** shortcut again?

```
[student@desktopX ~]$ head <Esc>.  
[student@desktopX ~]$ head /usr/bin/clean-binary-files  
#!/bin/sh  
#  
# Script to clean binary files.  
#  
# JPackage Project <http://www.jpackage.org/>  
#  
# $Id: clean-binary-files,v 1.1 2006/09/19 19:39:37 fnasser Exp $  
  
# Import java functions  
[ -r "/usr/share/java-utils/java-functions" ] \
```

9. Display the last 10 lines at the bottom of the **/usr/bin/clean-binary-files** file.

Use the **tail** command.

```
[student@desktopX ~]$ tail <Esc>.  
[student@desktopX ~]$ tail /usr/bin/clean-binary-files  
...Output omitted...
```

10. Repeat the previous command, but use the **-n 20** option to display the last 20 lines in the file. Use command line editing to accomplish this with a minimal amount of keystrokes.

Up Arrow displays the previous command. **Ctrl+a** makes the cursor jump to the beginning of the line. **Ctrl+Right Arrow** jumps to the next word, then add the **-n 20** option and hit **Enter** to execute the command.

```
[student@desktopX ~]$ tail -n 20 /usr/bin/clean-binary-files  
...Output omitted...
```

11. Execute the **date** command without any arguments to display the current date and time.

```
[student@desktopX ~]$ date
```

```
Thu Apr 3 10:48:30 PDT 2014
```

12. Use **bash** history to display just the time.

Display the list of previous commands with the **history** command to identify the specific **date** command to be executed. Execute the command with the **!number** history command.

```
[student@desktopX ~]$ history
...
44 date +%X
...
[student@desktopX ~]$ !44
date +%X
10:49:56 AM
```

13. Finish your session with the **bash** shell.

Use either **exit** or the **Ctrl+d** key combination to close the shell.

```
[student@desktopX ~]$ exit
```

Summary

Accessing the Command Line Using the Local Console

Use the physical console to view output and input commands with correct syntax using the **bash** shell.

Accessing the Command Line Using the Desktop

Use the GNOME graphical environment to launch applications, especially the graphical terminal program.

Executing Commands Using the Bash Shell

Use the tab completion, command history, and command line-editing features of the Bash shell to execute commands more efficiently.



CHAPTER 2

MANAGING FILES FROM THE COMMAND LINE

Overview	
Goal	To copy, move, create, delete, and organize files while working from the Bash shell prompt.
Objectives	<ul style="list-style-type: none">Identify the purpose for important directories on a Linux system.Specify files using absolute and relative path names.Create, copy, move, and remove files and directories using command-line utilities.Match one or more file names using shell expansion as arguments to shell commands.
Sections	<ul style="list-style-type: none">The Linux File System Hierarchy (and Practice)Locating Files by Name (and Practice)Managing Files Using Command-Line Tools (and Practice)Matching File Names Using Path Name Expansion (and Practice)
Lab	<ul style="list-style-type: none">Managing Files with Shell Expansion

The Linux File System Hierarchy

Objectives

After completing this section, students should be able to understand fundamental file system layout, organization, and the location of key file types.

The file system hierarchy

All files on a Linux system are stored on file systems which are organized into a single *inverted* tree of directories, known as a *file system hierarchy*. This tree is inverted because the root of the tree is said to be at the *top* of the hierarchy, and the branches of directories and subdirectories stretch *below* the root.

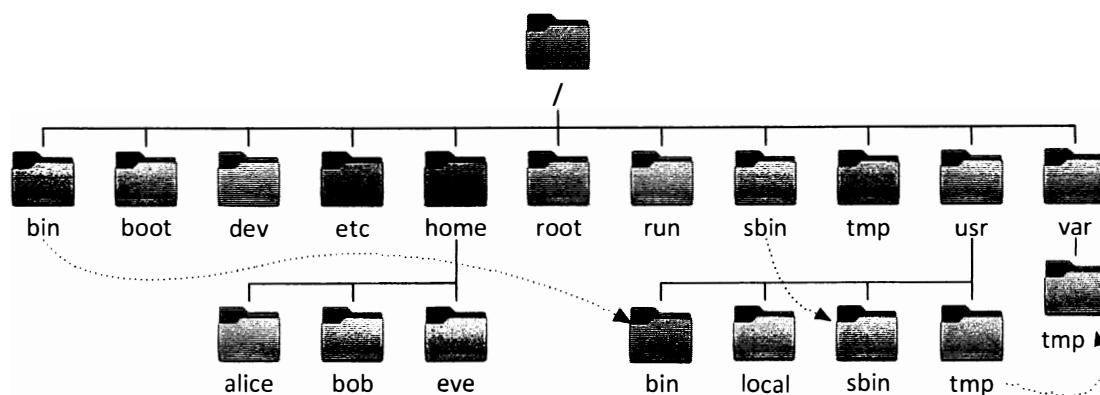


Figure 2.1: Significant file system directories in Red Hat Enterprise Linux 7

The directory `/` is the root directory at the top of the file system hierarchy. The `/` character is also used as a *directory separator* in file names. For example, if `etc` is a subdirectory of the `/` directory, we could call that directory `/etc`. Likewise, if the `/etc` directory contained a file named `issue`, we could refer to that file as `/etc/issue`.

Subdirectories of `/` are used for standardized purposes to organize files by type and purpose. This makes it easier to find files. For example, in the root directory, the subdirectory `/boot` is used for storing files needed to boot the system.



Note

The following terms are encountered in describing file system directory contents:

- *static* is content that remains unchanged until explicitly edited or reconfigured.
- *dynamic* or *variable* is content typically modified or appended by active processes.
- *persistent* is content, particularly configuration settings, that remain after a reboot.
- *runtime* is process- or system-specific content or attributes cleared during reboot.

The following table lists some of the most important directories on the system by name and purpose.

Important Red Hat Enterprise Linux directories

Location	Purpose
/usr	Installed software, shared libraries, include files, and static read-only program data. Important subdirectories include: - /usr/bin : <i>User commands</i> . - /usr/sbin : <i>System administration commands</i> . - /usr/local : <i>Locally customized software</i> .
/etc	Configuration files specific to this system.
/var	Variable data specific to this system that should persist between boots. Files that dynamically change (e.g. databases, cache directories, log files, printer-spooled documents, and website content) may be found under /var .
/run	Runtime data for processes started since the last boot. This includes process ID files and lock files, among other things. The contents of this directory are recreated on reboot. (This directory consolidates /var/run and /var/lock from older versions of Red Hat Enterprise Linux.)
/home	<i>Home directories</i> where regular users store their personal data and configuration files.
/root	Home directory for the administrative superuser, root.
/tmp	A world-writable space for temporary files. Files which are more than 10 days old are deleted from this directory automatically. Another temporary directory exists, /var/tmp , in which files that have not been accessed, changed, or modified in more than 30 days are deleted automatically.
/boot	Files needed in order to start the boot process.
/dev	Contains special <i>device files</i> which are used by the system to access hardware.

**Important**

In Red Hat Enterprise Linux 7, four older directories in **/** now have identical contents as their counterparts located in **/usr**:

- **/bin** and **/usr/bin**.
- **/sbin** and **/usr/sbin**.
- **/lib** and **/usr/lib**.
- **/lib64** and **/usr/lib64**.

In older versions of Red Hat Enterprise Linux, these were distinct directories containing different sets of files. In RHEL 7, the directories in **/** are symbolic links to the matching directories in **/usr**.



References

hier(7) man page

Filesystem Hierarchy Standard
<http://www.pathname.com/fhs>

Practice: File System Hierarchy

Quiz

Match the following items to their counterparts in the table.

/	/etc	/home	/root	/run	/tmp	/usr
	/usr/bin	/usr/sbin	/var			

Directory purpose	Location
This directory contains static, persistent system configuration data.	
This is the system's root directory.	
User home directories are located under this directory.	
This is the root account's home directory.	
This directory contains dynamic configuration data, such as FTP and websites.	
Regular user commands and utilities are located here.	
System administration binaries, for root use, are here.	
Temporary files are stored here.	

Directory purpose	Location
Contains dynamic, non-persistent application runtime data.	
Contains installed software programs and libraries.	

Solution

Match the following items to their counterparts in the table.

Directory purpose	Location
This directory contains static, persistent system configuration data.	/etc
This is the system's root directory.	/
User home directories are located under this directory.	/home
This is the root account's home directory.	/root
This directory contains dynamic configuration data, such as FTP and websites.	/var
Regular user commands and utilities are located here.	/usr/bin
System administration binaries, for root use, are here.	/usr/sbin
Temporary files are stored here.	/tmp
Contains dynamic, non-persistent application runtime data.	/run
Contains installed software programs and libraries.	/usr

Locating Files by Name

Objectives

After completing this section, students should be able to correctly use absolute path names, change a working directory, and use commands to determine directory locations and contents.

Absolute paths and relative paths

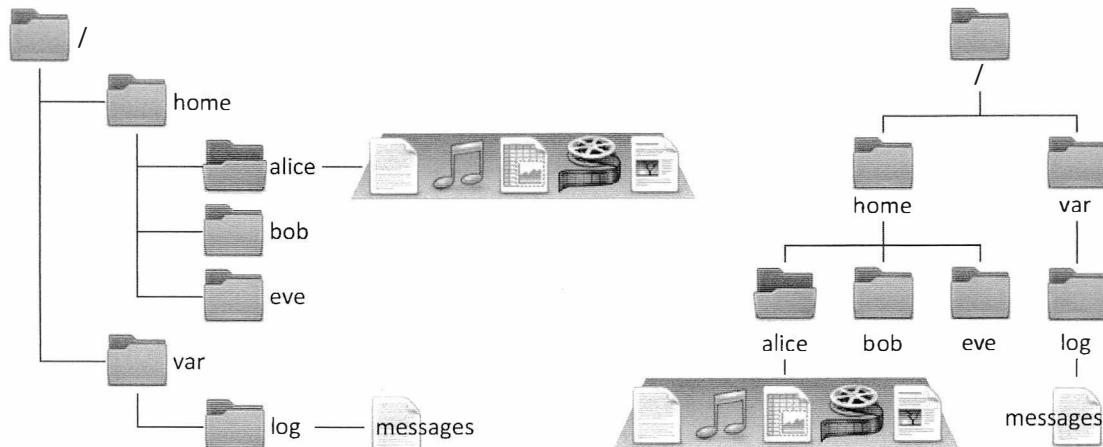


Figure 2.2: The common file browser view (left) is equivalent to the top-down view (right)

The *path* of a file or directory specifies its unique file system location. Following a file path traverses one or more named subdirectories, delimited by a forward slash (/), until the destination is reached. Standard file behavior definitions apply to directories (also called *folders*) the same as other file types.



Important

Although a **Space** is an acceptable character in Linux file names, a space is the delimiter used by the command shell for command syntax interpretation. New administrators are advised to avoid using spaces in file names, since file names that include spaces frequently result in undesired command execution behavior.

Absolute paths

An *absolute path* is a *fully qualified* name, beginning at the root (/) directory and specifying each subdirectory traversed to reach and uniquely represent a single file. Every file in a file system has a unique absolute path name, recognized with a simple rule: A path name with a forward slash (/) as the first character is an absolute path name. For example, the absolute path name for the system message log file is **/var/log/messages**. Absolute path names can be long to type, so files may also be located *relatively*.

When a user logs in and opens a command window, the initial location is normally the user's home directory. System processes also have an initial directory. Users and processes navigate to other directories as needed; the terms *working directory* or *current working directory* refer to their *current* location.

Relative paths

Like an absolute path, a *relative path* identifies a unique file, specifying only the path necessary to reach the file from the working directory. Recognizing relative path names follows a simple rule: A path name with *anything other than* a forward slash (/) as a first character is a relative path name. A user in the /var directory could refer to the message log file relatively as **log/messages**.

For standard Linux file systems, the path name of a file, including all / characters, may be no more than 4095 bytes long. Each component of the path name separated by / characters may be no more than 255 bytes long. File names can use any UTF-8 encoded Unicode character except / and the **NUL** character. (ASCII characters require one byte; other Latin, Greek, Hebrew, or Cyrillic characters take two bytes; remaining characters in the Unicode Basic Multilingual Plane take three; and no character will take more than four bytes.)

Linux file systems—including, but not limited to, ext4, XFS, BTRFS, GFS2, and GlusterFS—are case-sensitive. Creating **FileCase.txt** and **filecase.txt** in the same directory results in two unique files. Although many non-Linux file systems are supported in Linux, each has unique file naming rules. For example, the ubiquitous VFAT file system is not case-sensitive and allows only one of the two example files to be created. However, VFAT, along with Microsoft's NTFS and Apple's HFS+, has *case preserving* behavior. Although these file systems are *not* case-sensitive (enforced primarily to support backward compatibility), they do display file names with the original capitalization used when the file was created.

Navigating paths

The **pwd** command displays the full path name of the current location, which helps determine appropriate syntax for reaching files using relative path names. The **ls** command lists directory contents for the specified directory or, if no directory is given, for the current directory.

```
[student@desktopX ~]$ pwd
/home/student
[student@desktopX ~]$ ls
Desktop Documents Downloads Music Pictures Public Templates Videos
[student@desktopX ~]$
```

Use the **cd** command to change directories. With a working directory of **/home/student**, relative path syntax is shortest to reach the **Videos** subdirectory. The **Documents** subdirectory is then reached using absolute path syntax.

```
[student@desktopX ~]$ cd Videos
[student@desktopX Videos]$ pwd
/home/student/Videos
[student@desktopX Videos]$ cd /home/student/Documents
[student@desktopX Documents]$ pwd
/home/student/Documents
[student@desktopX Documents]$ cd
[student@desktopX ~]$ pwd
/home/student
[student@desktopX ~]$
```

The shell program prompt displays, for brevity, only the last component of the current directory path. For **/home/student/Videos**, only **Videos** displays. At any time, return to the user's home directory using **cd** without specifying a destination. The prompt displays the *tilde* (~) character when the user's current directory is their home directory.

The **touch** command normally updates a file's timestamp to the current date and time without otherwise modifying it. This is useful for creating empty files, which can be used for practice, since "touching" a file name that does not exist causes the file to be created. Using **touch**, practice files are created in the **Documents** and **Videos** subdirectories.

```
[student@desktopX ~]$ touch Videos/blockbuster1.ogg
[student@desktopX ~]$ touch Videos/blockbuster2.ogg
[student@desktopX ~]$ touch Documents/thesis_chapter1.odf
[student@desktopX ~]$ touch Documents/thesis_chapter2.odf
[student@desktopX ~]$
```

The **ls** command has multiple options for displaying attributes on files. The most common and useful are **-l** (long listing format), **-a** (all files, includes *hidden* files), and **-R** (recursive, to include the contents of all subdirectories).

```
[student@desktopX ~]$ ls -l
total 15
drwxr-xr-x. 2 student student 4096 Feb  7 14:02 Desktop
drwxr-xr-x. 2 student student 4096 Jan  9 15:00 Documents
drwxr-xr-x. 3 student student 4096 Jan  9 15:00 Downloads
drwxr-xr-x. 2 student student 4096 Jan  9 15:00 Music
drwxr-xr-x. 2 student student 4096 Jan  9 15:00 Pictures
drwxr-xr-x. 2 student student 4096 Jan  9 15:00 Public
drwxr-xr-x. 2 student student 4096 Jan  9 15:00 Templates
drwxr-xr-x. 2 student student 4096 Jan  9 15:00 Videos
[student@desktopX ~]$ ls -a
total 15
drwx----- 16 student student 4096 Feb  8 16:15 .
drwxr-xr-x.  6 root      root    4096 Feb  8 16:13 ..
-rw-----  1 student student 22664 Feb  8 00:37 .bash_history
-rw-r--r--.  1 student student   18 Jul  9 2013 .bash_logout
-rw-r--r--.  1 student student  176 Jul  9 2013 .bash_profile
-rw-r--r--.  1 student student  124 Jul  9 2013 .bashrc
drwxr-xr-x.  4 student student 4096 Jan 20 14:02 .cache
drwxr-xr-x.  8 student student 4096 Feb  5 11:45 .config
drwxr-xr-x.  2 student student 4096 Feb  7 14:02 Desktop
drwxr-xr-x.  2 student student 4096 Jan  9 15:00 Documents
drwxr-xr-x.  3 student student 4096 Jan 25 20:48 Downloads
drwxr-xr-x. 11 student student 4096 Feb  6 13:07 .gnome2
drwx-----  2 student student 4096 Jan 20 14:02 .gnome2_private
-rw-----  1 student student 15190 Feb  8 09:49 .ICEauthority
drwxr-xr-x.  3 student student 4096 Jan  9 15:00 .local
drwxr-xr-x.  2 student student 4096 Jan  9 15:00 Music
drwxr-xr-x.  2 student student 4096 Jan  9 15:00 Pictures
drwxr-xr-x.  2 student student 4096 Jan  9 15:00 Public
drwxr-xr-x.  2 student student 4096 Jan  9 15:00 Templates
drwxr-xr-x.  2 student student 4096 Jan  9 15:00 Videos
[student@desktopX ~]$
```

The two special directories at the top of the listing refer to the current directory (.) and the *parent* directory (..). These special directories exist in every directory on the system. Their usefulness will become apparent when file management commands are practiced.



Important

File names beginning with a dot (.) indicate files *hidden* from normal view using **ls** and other commands. This is *not* a security feature. Hidden files keep necessary user configuration files from cluttering home directories. Many commands process hidden files only with specific command-line options, preventing one user's configuration from being accidentally copied to other directories or users.

To protect file *contents* from improper viewing requires the use of *file permissions*.

```
[student@desktopX ~]$ ls -R
.:
Desktop Documents Downloads Music Pictures Public Templates Videos

./Desktop:

./Documents:
thesis_chapter1.odf thesis_chapter2.odf

./Downloads:

./Music:

./Pictures:

./Public:

./Templates:

./Videos:
blockbuster1.ogg blockbuster2.ogg
[student@desktopX ~]$
```

The **cd** command has many options. A few are so useful as to be worth practicing early and using often. The command **cd -** changes directory to the directory where the user was *previous* to the current directory. Watch as this user takes advantage of this behavior to alternate between two directories, useful when processing a series of similar tasks.

```
[student@desktopX ~]$ cd Videos
[student@desktopX Videos]$ pwd
/home/student/Videos
[student@desktopX Videos]$ cd /home/student/Documents
[student@desktopX Documents]$ pwd
/home/student/Documents
[student@desktopX Documents]$ cd -
[student@desktopX Videos]$ pwd
/home/student/Videos
[student@desktopX Videos]$ cd -
[student@desktopX Documents]$ pwd
/home/student/Documents
[student@desktopX Documents]$ cd -
[student@desktopX Videos]$ pwd
/home/student/Videos
[student@desktopX Videos]$ cd
[student@desktopX ~]$
```

The **cd ..** command uses the **..** hidden directory to move up one level to the *parent* directory, without needing to know the exact parent name. The other hidden directory **(.)** specifies the *current directory* on commands in which the current location is either the source or destination argument, avoiding the need to type out the directory's absolute path name.

```
[student@desktopX Videos]$ pwd  
/home/student/Videos  
[student@desktopX Videos]$ cd .  
[student@desktopX Videos]$ pwd  
/home/student/Videos  
[student@desktopX Videos]$ cd ..  
[student@desktopX ~]$ pwd  
/home/student  
[student@desktopX ~]$ cd ..  
[student@desktopX home]$ pwd  
/home  
[student@desktopX home]$ cd ..  
[student@desktopX /]$ pwd  
/  
[student@desktopX /]$ cd  
[student@desktopX ~]$ pwd  
/home/student  
[student@desktopX ~]$
```

R References

info libc 'file name resolution' (*GNU C Library Reference Manual*)

- Section 11.2.2 File name resolution

bash(1), cd(1), ls(1), pwd(1), unicode(7), and utf-8(7) man pages

UTF-8 and Unicode

<http://www.utf-8.com/>

Practice: Locating Files and Directories

Quiz

Match the following items to their counterparts in the table.

cd	cd -	cd ..	cd ../../	cd /bin	cd bin	ls -al
ls -l ~		pwd				

Action to accomplish	Command
List the current user's home directory (long format) in simplest syntax, when it is not the current location.	
Return to the current user's home directory.	
Determine the absolute path name of the current location.	
Return to the most previous working directory.	
Move up two levels from the current location.	
List the current location (long format) with hidden files.	
Move to the binaries location, from any current location.	
Move up to the parent of the current location.	
Move to the binaries location, from the root directory.	

Solution

Match the following items to their counterparts in the table.

Action to accomplish	Command
List the current user's home directory (long format) in simplest syntax, when it is not the current location.	ls -l ~
Return to the current user's home directory.	cd
Determine the absolute path name of the current location.	pwd
Return to the most previous working directory.	cd -
Move up two levels from the current location.	cd ../../..
List the current location (long format) with hidden files.	ls -al
Move to the binaries location, from any current location.	cd /bin
Move up to the parent of the current location.	cd ..
Move to the binaries location, from the root directory.	cd bin

Managing Files Using Command-Line Tools

Objectives

After completing this section, students should be able to create, copy, link, move, and remove files and subdirectories in various directories.

Command-line file management

File management involves creating, deleting, copying, and moving files. Additionally, directories can be created, deleted, copied, and moved to help organize files logically. When working at the command line, file management requires awareness of the current working directory to choose either absolute or relative path syntax as most efficient for the immediate task.

File management commands

Activity	Single source ^(note)	Multiple source ^(note)
Copy file	cp file1 file2	cp file1 file2 file3 dir ⁽⁵⁾
Move file	mv file1 file2 ⁽¹⁾	mv file1 file2 file3 dir ⁽⁴⁾
Remove file	rm file1	rm -f file1 file2 file3 ⁽⁵⁾
Create directory	mkdir dir	mkdir -p par1/par2/dir ⁽⁶⁾
Copy directory	cp -r dir1 dir2 ⁽²⁾	cp -r dir1 dir2 dir3 dir4 ⁽⁴⁾
Move directory	mv dir1 dir2 ⁽³⁾	mv dir1 dir2 dir3 dir4 ⁽⁴⁾
Remove directory	rm -r dir1 ⁽²⁾	rm -rf dir1 dir2 dir3 ⁽⁵⁾
Note:	⁽¹⁾ The result is a rename. ⁽²⁾ The "recursive" option is required to process a source directory. ⁽³⁾ If dir2 exists, the result is a move. If dir2 doesn't exist, the result is a rename. ⁽⁴⁾ The last argument must be a directory. ⁽⁵⁾ Use caution with "force" option; you will not be prompted to confirm your action. ⁽⁶⁾ Use caution with "create parent" option; typing errors are not caught.	

Create directories

The **mkdir** command creates one or more directories or subdirectories, generating errors if the file name already exists or when attempting to create a directory in a parent directory that doesn't exist. The **-p parent** option creates missing parent directories for the requested destination. Be cautious when using **mkdir -p**, since accidental spelling mistakes create unintended directories without generating error messages.

```
[student@desktopX ~]$ mkdir Video/Watched
mkdir: cannot create directory `Video/Watched': No such file or directory
```

The **mkdir** failed because **Videos** was misspelled. "**Video**" does not exist as a location in which to create the **Watched** subdirectory. If a **-p** were used, the user would not have received an error message and now have two directories, **Video** and **Videos**.

```
[student@desktopX ~]$ mkdir Videos/Watched
[student@desktopX ~]$ cd Documents
```

```
[student@desktopX Documents]$ mkdir ProjectX ProjectY
[student@desktopX Documents]$ mkdir -p Thesis/Chapter1 Thesis/Chapter2 Thesis/Chapter3
[student@desktopX Documents]$ cd
[student@desktopX ~]$ ls -R Videos Documents
Documents:
ProjectX ProjectY Thesis thesis_chapter1.odf thesis_chapter2.odf

Documents/ProjectX:

Documents/ProjectY:

Documents/Thesis:
Chapter1 Chapter2 Chapter3

Documents/Thesis/Chapter1:

Documents/Thesis/Chapter2:

Documents/Thesis/Chapter3:

Videos:
blockbuster1.ogg blockbuster2.ogg Watched

Videos/Watched:

[student@desktopX ~]$
```

The last **mkdir** created three Chapter*N* subdirectories with one command. The **-p parent** option created the missing parent directory **Thesis**.

Copy files

The **cp** command copies one or more files to become new, independent files. Syntax allows copying an existing file to a new file in the current or another directory, or copying multiple files into another directory. In any destination, new file names must be unique. If the new file name is not unique, the copy command will overwrite the existing file.

```
[student@desktopX ~]$ cd Videos
[student@desktopX Videos]$ cp blockbuster1.ogg blockbuster3.ogg
[student@desktopX Videos]$ ls -l
total 0
-rw-rw-r-- 1 student student 0 Feb 8 16:23 blockbuster1.ogg
-rw-rw-r-- 1 student student 0 Feb 8 16:24 blockbuster2.ogg
-rw-rw-r-- 1 student student 0 Feb 8 19:02 blockbuster3.ogg
drwxrwxr-x. 2 student student 4096 Feb 8 23:35 Watched
[student@desktopX Videos]$
```

When copying multiple files with one command, the last argument must be a directory. Copied files retain their original names in the new directory. Conflicting file names that exist at a destination may be overwritten. To protect users from accidentally overwriting directories with contents, multiple file **cp** commands ignore directories specified as a source. Copying non-empty directories, with contents, requires the **-r recursive** option.

```
[student@desktopX Videos]$ cd ../Documents
[student@desktopX Documents]$ cp thesis_chapter1.odf thesis_chapter2.odf Thesis ProjectX
cp: omitting directory `Thesis'
[student@desktopX Documents]$ cp -r Thesis ProjectX
[student@desktopX Documents]$ cp thesis_chapter2.odf Thesis/Chapter2/
[student@desktopX Documents]$ ls -R
.:
```

```

ProjectX ProjectY Thesis thesis_chapter1.odf thesis_chapter2.odf

./ProjectX:
Thesis thesis_chapter1.odf thesis_chapter2.odf

./ProjectX/Thesis:

./ProjectY:

./Thesis:
Chapter1 Chapter2 Chapter3

./Thesis/Chapter1:

./Thesis/Chapter2:
thesis_chapter2.odf

./Thesis/Chapter3:
[student@desktopX Documents]$
```

In the first **cp** command, **Thesis** failed to copy, but **thesis_chapter1.odf** and **thesis_chapter2.odf** succeeded. Using the **-r recursive** option, copying **Thesis** succeeded.

Move files

The **mv** command renames files in the same directory, or relocates files to a new directory. File contents remain unchanged. Files moved to a different file system require creating a new file by copying the source file, then deleting the source file. Although normally transparent to the user, large files may take noticeably longer to move.

```

[student@desktopX Videos]$ cd ../Documents
[student@desktopX Documents]$ ls -l
total 0
-rw-rw-r--. 1 student student    0 Feb  8 16:24 thesis_chapter1.odf
-rw-rw-r--. 1 student student    0 Feb  8 16:24 thesis_chapter2.odf
[student@desktopX Documents]$ mv thesis_chapter2.odf thesis_chapter2_reviewed.odf
[student@desktopX Documents]$ mv thesis_chapter1.odf Thesis/Chapter1
[student@desktopX Documents]$ ls -lR
.:
total 16
drwxrwxr-x. 2 student student 4096 Feb 11 11:58 ProjectX
drwxrwxr-x. 2 student student 4096 Feb 11 11:55 ProjectY
drwxrwxr-x. 5 student student 4096 Feb 11 11:56 Thesis
-rw-rw-r--. 1 student student    0 Feb 11 11:54 thesis_chapter2_reviewed.odf

./ProjectX:
total 0
-rw-rw-r--. 1 student student 0 Feb 11 11:58 thesis_chapter1.odf
-rw-rw-r--. 1 student student 0 Feb 11 11:58 thesis_chapter2.odf

./ProjectX/Thesis:
total 0

./ProjectY:
total 0

./Thesis:
total 12
drwxrwxr-x. 2 student student 4096 Feb 11 11:59 Chapter1
drwxrwxr-x. 2 student student 4096 Feb 11 11:56 Chapter2
drwxrwxr-x. 2 student student 4096 Feb 11 11:56 Chapter3

./Thesis/Chapter1:
```

```
total 0
-rw-rw-r-- 1 student student 0 Feb 11 11:54 thesis_chapter1.odf

./Thesis/Chapter2:
total 0
-rw-rw-r-- 1 student student 0 Feb 11 11:54 thesis_chapter2.odf

./Thesis/Chapter3:
total 0
[student@desktopX Documents]$
```

The first **mv** command is an example of renaming a file. The second causes the file to be relocated to another directory.

Remove files and directories

Default syntax for **rm** deletes files, but not directories. Deleting a directory, and potentially many subdirectories and files below it, requires the **-r recursive** option. There is no command-line undelete feature, nor a trash bin from which to restore.

```
[student@desktopX Documents]$ pwd
/home/student/Documents
[student@desktopX Documents]$ rm thesis_chapter2_reviewed.odf
[student@desktopX Documents]$ rm Thesis/Chapter1
rm: cannot remove `Thesis/Chapter1': Is a directory
[student@desktopX Documents]$ rm -r Thesis/Chapter1
[student@desktopX Documents]$ ls -l Thesis
total 8
drwxrwxr-x. 2 student student 4096 Feb 11 12:47 Chapter2
drwxrwxr-x. 2 student student 4096 Feb 11 12:48 Chapter3
[student@desktopX Documents]$ rm -ri Thesis
rm: descend into directory `Thesis'? y
rm: descend into directory `Thesis/Chapter2'? y
rm: remove regular empty file `Thesis/Chapter2/thesis_chapter2.odf'? y
rm: remove directory `Thesis/Chapter2'? y
rm: remove directory `Thesis/Chapter3'? y
rm: remove directory `Thesis'? y
[student@desktopX Documents]$
```

After **rm** failed to delete the **Chapter1** directory, the **-r recursive** option succeeded. The last **rm** command parsed into each subdirectory first, individually deleting contained files before removing each now-empty directory. Using **-i** will interactively prompt for each deletion. This is essentially the opposite of **-f** which will force the deletion without prompting the user.

The **rmdir** command deletes directories only if empty. Removed directories cannot be undeleted.

```
[student@desktopX Documents]$ pwd
/home/student/Documents
[student@desktopX Documents]$ rmdir ProjectY
[student@desktopX Documents]$ rmdir ProjectX
rmdir: failed to remove `ProjectX': Directory not empty
[student@desktopX Documents]$ rm -r ProjectX
[student@desktopX Documents]$ ls -lR
.:
total 0
[student@desktopX Documents]$
```

The **rmdir** command failed to delete non-empty **ProjectX**, but **rm -r** succeeded.



References

cp(1), mkdir(1), mv(1), rm(1), and rmdir(1) man pages

Practice: Command-Line File Management

Guided exercise

In this lab, you will practice efficient techniques for creating and organizing files using directories, file copies, and links.

Outcomes:

Students will practice creating, rearranging, and deleting files.

Before you begin...

Log into your student account on serverX. Begin in your home directory.

- 1. In your home directory, create sets of empty practice files to use for the remainder of this lab. If the intended command is not immediately recognized, students are expected to use the guided solution to see and practice how the task is accomplished. Use the shell tab completion to locate and complete path names more easily.

Create six files with names of the form **songX.mp3**.

Create six files with names of the form **snapX.jpg**.

Create six files with names of the form **filmX.avi**.

In each set, replace X with the numbers 1 through 6.

```
[student@serverX ~]$ touch song1.mp3 song2.mp3 song3.mp3 song4.mp3 song5.mp3  
song6.mp3  
[student@serverX ~]$ touch snap1.jpg snap2.jpg snap3.jpg snap4.jpg snap5.jpg  
snap6.jpg  
[student@serverX ~]$ touch film1.avi film2.avi film3.avi film4.avi film5.avi  
film6.avi  
[student@serverX ~]$ ls -l
```

- 2. From your home directory, move the song files into your **Music** subdirectory, the snapshot files into your **Pictures** subdirectory, and the movie files into your **Videos** subdirectory.

When distributing files from one location to many locations, first change to the directory containing the *source* files. Use the simplest path syntax, absolute or relative, to reach the destination for each file management task.

```
[student@serverX ~]$ mv song1.mp3 song2.mp3 song3.mp3 song4.mp3 song5.mp3  
song6.mp3 Music  
[student@serverX ~]$ mv snap1.jpg snap2.jpg snap3.jpg snap4.jpg snap5.jpg  
snap6.jpg Pictures  
[student@serverX ~]$ mv film1.avi film2.avi film3.avi film4.avi film5.avi  
film6.avi Videos  
[student@serverX ~]$ ls -l Music Pictures Videos
```

- 3. In your home directory, create three subdirectories for organizing your files into projects. Call these directories **friends**, **family**, and **work**. Create all three with one command.

You will use these directories to rearrange your files into projects.

```
[student@serverX ~]$ mkdir friends family work
[student@serverX ~]$ ls -l
```

- 4. You will collect some of the new files into the project directories for family and friends. Use as many commands as needed. You do not have to use only one command as in the example. For each project, first change to the project directory, then copy the source files into this directory. You are making copies, since you will keep the originals after giving these projects to family and friends.

Copy files (all types) containing numbers 1 and 2 to the friends folder.

Copy files (all types) containing numbers 3 and 4 to the family folder.

When collecting files from multiple locations into one location, change to the directory that will contain the *destination* files. Use the simplest path syntax, absolute or relative, to reach the source for each file management task.

```
[student@serverX ~]$ cd friends
[student@serverX friends]$ cp ~/Music/song1.mp3 ~/Music/song2.mp3 ~/Pictures/
snap1.jpg ~/Pictures/snap2.jpg ~/Videos/film1.avi ~/Videos/film2.avi .
[student@serverX friends]$ ls -l
[student@serverX friends]$ cd ../../family
[student@serverX family]$ cp ~/Music/song3.mp3 ~/Music/song4.mp3 ~/Pictures/
snap3.jpg ~/Pictures/snap4.jpg ~/Videos/film3.avi ~/Videos/film4.avi .
[student@serverX family]$ ls -l
```

- 5. For your work project, you will create additional copies.

```
[student@serverX family]$ cd ../../work
[student@serverX work]$ cp ~/Music/song5.mp3 ~/Music/song6.mp3 ~/Pictures/
snap5.jpg ~/Pictures/snap6.jpg ~/Videos/film5.avi ~/Videos/film6.avi .
[student@serverX work]$ ls -l
```

- 6. Your projects are now done. Time to clean up the projects.

Change to your home directory. Attempt to delete both the family and friends projects with a single **rmdir** command.

```
[student@serverX work]$ cd
[student@serverX ~]$ rmdir family friends
rmdir: failed to remove `family': Directory not empty
rmdir: failed to remove `friends': Directory not empty
```

Using the **rmdir** command should fail since both directories are non-empty.

- 7. Use another command that will succeed in deleting both the family and friends folders.

```
[student@serverX ~]$ rm -r family friends
[student@serverX ~]$ ls -l
```

- 8. Delete all the files in the work project, but do not delete the work directory.

```
[student@serverX ~]$ cd work
[student@serverX work]$ rm song5.mp3 song6.mp3 snap5.jpg snap6.jpg film5.avi
[student@serverX work]$ ls -l
```

- 9. Finally, from your home directory, use the **rmdir** command to delete the work directory. The command should succeed now that it is empty.

```
[student@serverX work]$ cd
[student@serverX ~]$ rmdir work
[student@serverX ~]$ ls -l
```

Matching File Names Using Path Name Expansion

Objectives

After completing this section, students should be able to use meta-characters and expansion techniques to improve file management processing efficiency.

File globbing: path name expansion

The Bash shell has a path name-matching capability historically called *globbing*, abbreviated from the “global command” file path expansion program of early UNIX. The Bash globbing feature, commonly called *pattern matching* or “wildcards”, makes managing large numbers of files easier. Using *meta-characters* that “expand” to match file and path names being sought, commands perform on a focused set of files at once.

Pattern matching

Globbing is a shell command-parsing operation that expands a wildcard pattern into a list of matching path names. Command-line meta-characters are replaced by the match list prior to command execution. Patterns, especially square-bracketed character classes, that do not return matches display the original pattern request as literal text. The following are common meta-characters and pattern classes.

Pattern	Matches
*	Any string of 0 or more characters.
?	Any single character.
~	The current user's home directory.
<code>~username</code>	User <code>username</code> 's home directory.
<code>~+</code>	The current working directory.
<code>~-</code>	The previous working directory.
<code>[abc...]</code>	Any one character in the enclosed class.
<code>[!abc...]</code>	Any one character <i>not</i> in the enclosed class.
<code>[^abc...]</code>	Any one character <i>not</i> in the enclosed class.
<code>[:alpha:]</code>	Any alphabetic character. ⁽¹⁾
<code>[:lower:]</code>	Any lower-case character. ⁽¹⁾
<code>[:upper:]</code>	Any upper-case character. ⁽¹⁾
<code>[:alnum:]</code>	Any alphabetic character or digit. ⁽¹⁾
<code>[:punct:]</code>	Any printable character not a space or alphanumeric. ⁽¹⁾
<code>[:digit:]</code>	Any digit, <code>0-9</code> . ⁽¹⁾
<code>[:space:]</code>	Any one whitespace character; may include tabs, newline, or carriage returns, and form feeds as well as space. ⁽¹⁾
Note	⁽¹⁾ pre-set POSIX character class; adjusts for current locale.

A sample set of files is useful to demonstrate expansion.

```
[student@desktopX ~]$ mkdir glob; cd glob
[student@desktopX glob]$ touch alfa bravo charlie delta echo able baker cast dog easy
[student@desktopX glob]$ ls
able alfa baker bravo cast charlie delta dog easy echo
[student@desktopX glob]$
```

First, simple pattern matches using * and ?.

```
[student@desktopX glob]$ ls a*
able alfa
[student@desktopX glob]$ ls *a*
able alfa baker bravo cast charlie delta easy
[student@desktopX glob]$ ls [ac]*
able alfa cast charlie
[student@desktopX glob]$ ls ???
able alfa cast easy echo
[student@desktopX glob]$ ls ??????
baker bravo delta
[student@desktopX glob]$
```

Tilde expansion

The tilde character (~), when followed by a slash delimiter, matches the current user's home directory. When followed by a string of characters up to a slash, it will be interpreted as a username, if one matches. If no username matches, then an actual tilde followed by the string of characters will be returned.

```
[student@desktopX glob]$ ls ~/glob
able alfa baker bravo cast charlie delta dog easy echo
[student@desktopX glob]$ echo ~/glob
/home/student/glob
[student@desktopX glob]$
```

Brace expansion

Brace expansion is used to generate discretionary strings of characters. Braces contain a comma-separated list of strings, or a sequence expression. The result includes the text preceding or following the brace definition. Brace expansions may be nested, one inside another.

```
[student@desktopX glob]$ echo {Sunday,Monday,Tuesday,Wednesday}.log
Sunday.log Monday.log Tuesday.log Wednesday.log
[student@desktopX glob]$ echo file{1..3}.txt
file1.txt file2.txt file3.txt
[student@desktopX glob]$ echo file{a..c}.txt
filea.txt fileb.txt filec.txt
[student@desktopX glob]$ echo file{a,b}{1,2}.txt
filea1.txt filea2.txt fileb1.txt fileb2.txt
[student@desktopX glob]$ echo file{a{1,2},b,c}.txt
filea1.txt filea2.txt fileb.txt filec.txt
[student@desktopX glob]$
```

Command substitution

Command substitution allows the output of a command to replace the command itself. Command substitution occurs when a command is enclosed with a beginning dollar sign and parenthesis, `$(command)`, or with backticks, ``command``. The form with backticks is older, and has two

disadvantages: 1) it can be easy to visually confuse backticks with single quote marks, and 2) backticks cannot be nested inside backticks. The `$(command)` form can nest multiple command expansions inside each other.

```
[student@desktopX glob]$ echo Today is `date +%A`.
Today is Wednesday.
[student@desktopX glob]$ echo The time is $(date +%M) minutes past $(date +%l%p).
The time is 26 minutes past 11AM.
[student@desktopX glob]$
```

Protecting arguments from expansion

Many characters have special meaning in the Bash shell. To ignore meta-character special meanings, *quoting* and *escaping* are used to protect them from shell expansion. The backslash (\) is an escape character in Bash, protecting the single following character from special interpretation. To protect longer character strings, single ('') or double quotes ("") are used to enclose strings.

Use double quotation marks to suppress globbing and shell expansion, but still allow command and variable substitution. Variable substitution is conceptually identical to command substitution, but may use optional brace syntax.

```
[student@desktopX glob]$ host=$(hostname); echo $host
desktopX
[student@desktopX glob]$ echo "***** hostname is ${host} *****"
***** hostname is desktopX *****
[student@desktopX glob]$ echo Your username variable is \$USER.
Your username variable is $USER.
[student@desktopX glob]$
```

Use single quotation marks to interpret *all* text literally. Observe the difference, on both screen and keyboard, between the single quote ('') and the command substitution backtick (`). Besides suppressing globbing and shell expansion, quotations direct the shell to additionally suppress command and variable substitution. The question mark is a meta-character that also needed protection from expansion.

```
[student@desktopX glob]$ echo "Will variable $host evaluate to $(hostname)?"
Will variable desktopX evaluate to desktopX?
[student@desktopX glob]$ echo 'Will variable $host evaluate to $(hostname)?'
Will variable $host evaluate to $(hostname)?
[student@desktopX glob]$
```

R

References

`bash(1)`, `cd(1)`, `glob(7)`, `isalpha(3)`, `ls(1)`, `path_resolution(7)`, and `pwd(1)` man pages

Practice: Path Name Expansion

Quiz

Match the following items to their counterparts in the table.

[[:digit:]]	*b	*b*	???*	[!b]*	[:upper:]]]*	b*
---------------	----	-----	------	-------	--------------	----

Requested match to find	Patterns
Only filenames beginning with "b"	
Only filenames ending in "b"	
Only filenames containing a "b"	
Only filenames where first character is not "b"	
Only filenames at least 3 characters in length	
Only filenames that contain a number	
Only filenames that begin with an upper-case letter	

Solution

Match the following items to their counterparts in the table.

Requested match to find	Patterns
Only filenames beginning with "b"	b*
Only filenames ending in "b"	*b
Only filenames containing a "b"	*b*
Only filenames where first character is not "b"	[!b]*
Only filenames at least 3 characters in length	???
Only filenames that contain a number	*[:digit:]*
Only filenames that begin with an upper-case letter	[:upper:]*

Lab: Managing Files with Shell Expansion

Performance checklist

In this lab, you will create, move, and remove files and folders using a variety of file name matching shortcuts.

Outcomes:

Familiarity and practice with many forms of wildcards for locating and using files.

Before you begin...

Perform the following steps on serverX unless directed otherwise. Log in as **student** and begin the lab in the home directory.

1. To begin, create sets of empty practice files to use in this lab. If an intended shell expansion shortcut is not immediately recognized, students are expected to use the solution to learn and practice. Use shell tab completion to locate file path names easily.

Create a total of 12 files with names **tv_seasonX_episodeY.ogg**. Replace X with the season number and Y with that season's episode, for two seasons of six episodes each.

2. As the author of a successful series of mystery novels, your next bestseller's chapters are being edited for publishing. Create a total of eight files with names **mystery_chapterX.odf**. Replace X with the numbers 1 through 8.
3. To organize the TV episodes, create two subdirectories named **season1** and **season2** under the existing **Videos** directory. Use one command.
4. Move the appropriate TV episodes into the season subdirectories. Use only two commands, specifying destinations using relative syntax.
5. To organize the mystery book chapters, create a two-level directory hierarchy with one command. Create **my_bestseller** under the existing **Documents** directory, and **chapters** beneath the new **my_bestseller** directory.
6. Using one command, create three more subdirectories directly under the **my_bestseller** directory. Name these subdirectories **editor**, **plot_change**, and **vacation**. The *create parent* option is not needed since the **my_bestseller** parent directory already exists.
7. Change to the **chapters** directory. Using the home directory shortcut to specify the source files, move all book chapters into the **chapters** directory, which is now your current directory. What is the simplest syntax to specify the destination directory?
8. The first two chapters are sent to the editor for review. To remember to not modify these chapters during the review, move those two chapters only to the **editor** directory. Use relative syntax starting from the **chapters** subdirectory.
9. Chapters 7 and 8 will be written while on vacation. Move the files from **chapters** to **vacation**. Use one command without wildcard characters.
10. With one command, change the working directory to the season 2 TV episodes location, then copy the first episode of the season to the **vacation** directory.

11. With one command, change the working directory to **vacation**, then list its files. Episode 2 is also needed. Return to the **season2** directory using the *previous working directory* shortcut. This will succeed if the last directory change was accomplished with one command. Copy the episode 2 file into **vacation**. Return to **vacation** using the shortcut again.
12. Chapters 5 and 6 may need a plot change. To prevent these changes from modifying original files, copy both files into **plot_change**. Move up one directory to **vacation**'s parent directory, then use one command from there.
13. To track changes, make three backups of chapter 5. Change to the **plot_change** directory. Copy **mystery_chapter5.odf** as a new file name to include the full date (Year-Mo-Da). Make another copy appending the current timestamp (as the number of seconds since the epoch) to ensure a unique file name. Also make a copy appending the current user (**\$USER**) to the file name. See the solution for the syntax of any you are unsure of (like what arguments to pass the **date**).

Note, we could also make the same backups of the chapter 6 files too.

14. The plot changes were not successful. Delete the **plot_change** directory. First, delete all of the files in the **plot_change** directory. Change directory up one level because the directory cannot be deleted while it is the working directory. Try to delete the directory using the **rm** command *without* the *recursive* option. This attempt should fail. Now use the **rmdir** command, which will succeed.
15. When the vacation is over, the **vacation** directory is no longer needed. Delete it using the **rm** command with the *recursive* option.

When finished, return to the home directory.

Solution

In this lab, you will create, move, and remove files and folders using a variety of file name matching shortcuts.

Outcomes:

Familiarity and practice with many forms of wildcards for locating and using files.

Before you begin...

Perform the following steps on serverX unless directed otherwise. Log in as **student** and begin the lab in the home directory.

1. To begin, create sets of empty practice files to use in this lab. If an intended shell expansion shortcut is not immediately recognized, students are expected to use the solution to learn and practice. Use shell tab completion to locate file path names easily.

Create a total of 12 files with names **tv_seasonX_episodeY.ogg**. Replace X with the season number and Y with that season's episode, for two seasons of six episodes each.

```
[student@serverX ~]$ touch tv_season{1..2}_episode{1..6}.ogg  
[student@serverX ~]$ ls -l
```

2. As the author of a successful series of mystery novels, your next bestseller's chapters are being edited for publishing. Create a total of eight files with names **mystery_chapterX.odf**. Replace X with the numbers 1 through 8.

```
[student@serverX ~]$ touch mystery_chapter{1..8}.odf  
[student@serverX ~]$ ls -l
```

3. To organize the TV episodes, create two subdirectories named **season1** and **season2** under the existing **Videos** directory. Use one command.

```
[student@serverX ~]$ mkdir Videos/season{1..2}  
[student@serverX ~]$ ls -lR
```

4. Move the appropriate TV episodes into the season subdirectories. Use only two commands, specifying destinations using relative syntax.

```
[student@serverX ~]$ mv tv_season1* Videos/season1  
[student@serverX ~]$ mv tv_season2* Videos/season2  
[student@serverX ~]$ ls -lR
```

5. To organize the mystery book chapters, create a two-level directory hierarchy with one command. Create **my_bestseller** under the existing **Documents** directory, and **chapters** beneath the new **my_bestseller** directory.

```
[student@serverX ~]$ mkdir -p Documents/my_bestseller/chapters  
[student@serverX ~]$ ls -lR
```

6. Using one command, create three more subdirectories directly under the **my_bestseller** directory. Name these subdirectories **editor**, **plot_change**, and **vacation**. The *create parent* option is not needed since the **my_bestseller** parent directory already exists.

```
[student@serverX ~]$ mkdir Documents/my_bestseller/{editor,plot_change,vacation}
[student@serverX ~]$ ls -lR
```

7. Change to the **chapters** directory. Using the home directory shortcut to specify the source files, move all book chapters into the **chapters** directory, which is now your current directory. What is the simplest syntax to specify the destination directory?

```
[student@serverX ~]$ cd Documents/my_bestseller/chapters
[student@serverX chapters]$ mv ~/mystery_chapter* .
[student@serverX chapters]$ ls -l
```

8. The first two chapters are sent to the editor for review. To remember to not modify these chapters during the review, move those two chapters only to the **editor** directory. Use relative syntax starting from the **chapters** subdirectory.

```
[student@serverX chapters]$ mv mystery_chapter1.odf mystery_chapter2.odf ../editor
[student@serverX chapters]$ ls -l
[student@serverX chapters]$ ls -l ../editor
```

9. Chapters 7 and 8 will be written while on vacation. Move the files from **chapters** to **vacation**. Use one command without wildcard characters.

```
[student@serverX chapters]$ mv mystery_chapter7.odf mystery_chapter8.odf ../vacation
[student@serverX chapters]$ ls -l
[student@serverX chapters]$ ls -l ../vacation
```

10. With one command, change the working directory to the season 2 TV episodes location, then copy the first episode of the season to the **vacation** directory.

```
[student@serverX chapters]$ cd ~/Videos/season2
[student@serverX season2]$ cp tv_season2_episode1.ogg ~/Documents/my_bestseller/
vacation
```

11. With one command, change the working directory to **vacation**, then list its files. Episode 2 is also needed. Return to the **season2** directory using the *previous working directory* shortcut. This will succeed if the last directory change was accomplished with one command. Copy the episode 2 file into **vacation**. Return to **vacation** using the shortcut again.

```
[student@serverX season2]$ cd ~/Documents/my_bestseller/vacation
[student@serverX vacation]$ ls -l
[student@serverX vacation]$ cd -
[student@serverX season2]$ cp tv_season2_episode2.ogg ~/Documents/my_bestseller/
vacation
[student@serverX vacation]$ cd -
[student@serverX vacation]$ ls -l
```

12. Chapters 5 and 6 may need a plot change. To prevent these changes from modifying original files, copy both files into **plot_change**. Move up one directory to **vacation**'s parent directory, then use one command from there.

```
[student@serverX vacation]$ cd ..  
[student@serverX my_bestseller]$ cp chapters/mystery_chapter[56].odf plot_change  
[student@serverX my_bestseller]$ ls -l chapters  
[student@serverX my_bestseller]$ ls -l plot_change
```

13. To track changes, make three backups of chapter 5. Change to the **plot_change** directory. Copy **mystery_chapter5.odf** as a new file name to include the full date (Year-Mo-Da). Make another copy appending the current timestamp (as the number of seconds since the epoch) to ensure a unique file name. Also make a copy appending the current user (**\$USER**) to the file name. See the solution for the syntax of any you are unsure of (like what arguments to pass the **date**).

```
[student@serverX my_bestseller]$ cd plot_change  
[student@serverX plot_change]$ cp mystery_chapter5.odf mystery_chapter5_$(date +  
%F).odf  
[student@serverX plot_change]$ cp mystery_chapter5.odf mystery_chapter5_$(date +  
%s).odf  
[student@serverX plot_change]$ cp mystery_chapter5.odf mystery_chapter5_${USER}.odf  
[student@serverX plot_change]$ ls -l
```

Note, we could also make the same backups of the chapter 6 files too.

14. The plot changes were not successful. Delete the **plot_change** directory. First, delete all of the files in the **plot_change** directory. Change directory up one level because the directory cannot be deleted while it is the working directory. Try to delete the directory using the **rm** command *without* the *recursive* option. This attempt should fail. Now use the **rmdir** command, which will succeed.

```
[student@serverX plot_change]$ rm mystery*  
[student@serverX plot_change]$ cd ..  
[student@serverX my_bestseller]$ rm plot_change  
rm: cannot remove 'plot_change': Is a directory  
[student@serverX my_bestseller]$ rmdir plot_change  
[student@serverX my_bestseller]$ ls -l
```

15. When the vacation is over, the **vacation** directory is no longer needed. Delete it using the **rm** command with the *recursive* option.

When finished, return to the home directory.

```
[student@serverX my_bestseller]$ rm -r vacation  
[student@serverX my_bestseller]$ ls -l  
[student@serverX my_bestseller]$ cd
```

Summary

The Linux File System Hierarchy

Identify the purpose for top-level directories in the Linux hierarchy.

Locating Files by Name

Interpret and appropriately use full and partial path file name syntax.

Managing Files Using Command-Line Tools

Work from the command line to create, move, and delete files and directories.

Matching File Names Using Path Name Expansion

Learn how to specify multiple files using many wildcard techniques.



CHAPTER 3

GETTING HELP IN RED HAT ENTERPRISE LINUX

Overview	
Goal	To resolve problems by using on-line help systems and Red Hat support utilities.
Objectives	<ul style="list-style-type: none">• Use the man Linux manual reader.• Use the pinfo GNU Info reader.• Use the Red Hat Package Manager (RPM) package documentation.• Use the redhat-support-tool command.
Sections	<ul style="list-style-type: none">• Reading Documentation Using man Command (and Practice)• Reading Documentation Using pinfo Command (and Practice)• Reading Documentation in /usr/share/doc (and Practice)• Getting Help From Red Hat (and Practice)
Lab	<ul style="list-style-type: none">• Viewing and Printing Help Documentation

Reading Documentation Using `man` Command

Objectives

After completing this section, students should be able to locate documentation and research answers about commands.

Introducing the `man` command

The historical Linux Programmer's Manual, from which man pages originate, was large enough to be multiple printed books. Each contained information for specific types of files, which have become the *sections* listed below. Articles are referred to as *topics*, as *pages* no longer applies.

Sections of the Linux manual

Section	Content type
1	User commands (<i>both executable and shell programs</i>)
2	System calls (<i>kernel routines invoked from user space</i>)
3	Library functions (<i>provided by program libraries</i>)
4	Special files (<i>such as device files</i>)
5	File formats (<i>for many configuration files and structures</i>)
6	Games (<i>historical section for amusing programs</i>)
7	Conventions, standards, and miscellaneous (<i>protocols, file systems</i>)
8	System administration and privileged commands (<i>maintenance tasks</i>)
9	Linux kernel API (<i>internal kernel calls</i>)



Note

Manual section 9 is a recent addition to Linux. Not all man section listings reference it.

To distinguish identical topic names in different sections, man page references include the section number in parentheses after the topic. For example, **passwd(1)** describes the command to change passwords, while **passwd(5)** explains the **/etc/passwd** file format for storing local user accounts.

To read specific man pages, use **man topic**. Topic contents display one screen at a time. Use arrow keys for single line scrolling or the space bar for the next screen. The **man** command searches manual sections in a configured order, displaying popular sections first. For example, **man passwd** displays **passwd(1)** by default. To display the man page topic from a specific section, include the section number argument: **man 5 passwd** displays **passwd(5)**.

Identify man pages by keyword

The ability to efficiently search for topics and navigate **man** pages is a critical administration skill. The following table lists basic **man** navigation commands:

Navigating man pages

Command	Result
Spacebar	Scroll forward (down) one screen
PageDown	Scroll forward (down) one screen
PageUp	Scroll backward (up) one screen
DownArrow	Scroll forward (down) one line
UpArrow	Scroll back (up) one line
d	Scroll forward (down) one half-screen
u	Scroll backward (up) one half-screen
/string	Search forward (down) for <i>string</i> in the man page
n	Repeat previous search forward (down) in the man page
N	Repeat previous search backward (up) in the man page
g	Go to start of the man page.
G	Go to end of the man page.
q	Exit man and return to the command shell prompt



Important

When performing searches, *string* allows *regular expression* syntax. While simple text (such as **passwd**) works as expected, regular expressions use meta-characters (such as \$, *, ., and ^) for more sophisticated pattern matching. Therefore, searching with strings which include program expression meta-characters, such as **make \$\$\$**, might yield unexpected results.

Regular expressions and syntax are discussed in *Red Hat System Administration II*, and in the **regex(7)** man topic.

Searching for man pages by keyword

A keyword search of man pages is performed using `man -k keyword`, which displays a list of keyword-matching man page topics with section numbers.

```
[student@desktopX ~]$ man -k passwd
checkPasswdAccess (3) - query the SELinux policy database in the kernel.
chpasswd (8)          - update passwords in batch mode
ckpasswd (8)           - nnrpd password authenticator
fgetpwent_r (3)        - get passwd file entry reentrantly
getpwent_r (3)         - get passwd file entry reentrantly
...
passwd (1)             - update user's authentication tokens
sslpasswd (1ssl)       - compute password hashes
passwd (5)              - password file
passwd.nntp (5)         - Passwords for connecting to remote NNTP servers
passwd2des (3)          - RFS password encryption
...
```

Popular system administration topics are in sections 1 (user commands), 5 (file formats), and 8 (administrative commands). Administrators using certain troubleshooting tools also use section 2 (system calls). The remaining sections are commonly for programmer reference or advanced administration.



Note

Keyword searches rely on an index generated by the `mandb(8)` command, which must be run as root. The command runs daily through `cron.daily`, or by `anacrontab` within an hour of boot if out of date.



Important

The `man` command `-K` option performs a full-text page search, not just titles and descriptions like the `-k`. A full-text search can use greater systems resources and take more time.



References

`man(1)`, `mandb(8)`, `man-pages(7)`, `less(1)`, `intro(1)`, `intro(2)`, `intro(5)`, `intro(7)`, `intro(8)` man pages

Practice: Using the man Command

Guided exercise

In this lab, you will practice finding relevant information by using **man** options and arguments.

Outcomes

Familiarity with the **man** Linux manual system and practice finding useful information by searching and browsing.

Before you begin...

Perform the following steps on serverX unless directed otherwise.

- 1. View the **gedit(1)** man page.

```
[student@serverX ~]$ man 1 gedit
```

- 2. Research how to edit a specific file using **gedit** from the command line.

gedit filename

- 3. Research the **gedit** option used to begin an editing session with the cursor at the end of the file.

gedit + filename

- 4. Research the **su(1)** man page.

```
[student@serverX ~]$ man 1 su
```

- 5. Research what **su** does when the *username* argument is omitted.

su assumes a *username* of **root**.

- 6. Research how **su** behaves when a single dash option is used.

su starts a child *login shell* (creating login environment by sourcing login scripts). Without the single dash, a non-login child shell is created, matching the user's current environment.

- 7. Consult the **passwd(1)** man page. Determine the options that will lock and unlock a user account when this command is used by **root**.

```
[student@serverX ~]$ man 1 passwd
```

passwd -l username

passwd -u username

- 8. Locate the two principles to remember according to the **passwd** man page authors. Search for the word "principle".

- Protect your password.
 - Choose a hard-to-guess password.
- 9. Consult the man page documenting the syntax of the **/etc/passwd** file. What is stored in the third field of each line?

The relevant man page is **passwd(5)**, found with **man -f passwd**.

The UID (numeric user ID) for each account.

- 10. Which command will list detailed information about a **zip** archive?

zipinfo(1) found with **man -k zip**

- 11. Which man page contains a list of parameters that can be passed to the kernel at boot time?

bootparam(7) found with **man -k boot**

- 12. Which command is used to tune **ext4** file system parameters?

tune2fs(8) found with **man -k ext4**

Reading Documentation Using pinfo Command

Objectives

After completing this section, students should be able to research answers using GNU Info documentation.

Introducing GNU info

Man pages have a formal format useful as a command reference, but less useful as general documentation. For such documents, the GNU Project developed a different online documentation system, known as GNU info. Info documents are an important resource on a Red Hat Enterprise Linux system because many fundamental components and utilities, such as the *coreutils* package and *glibc* standard libraries, are either developed by the GNU Project or utilize the info document system.

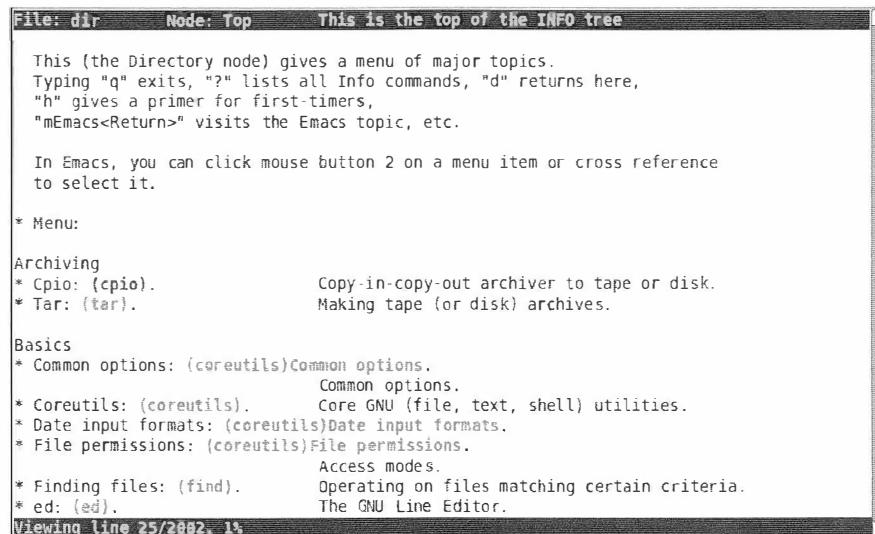


Figure 3.1: pinfo Info document viewer, top directory

Info documentation is structured as hyperlinked info nodes. This format is more flexible than man pages, allowing thorough discussion of complex commands and concepts. Like man pages, info nodes are read from the command line, using either the **info** or **pinfo** commands.

Some commands and utilities have both **man** pages and info documentation; usually, the info documentation will be more in-depth. Compare the differences in **tar** documentation using **man** and **pinfo**:

```
[student@desktopX ~]$ man tar
[student@desktopX ~]$ pinfo tar
```

The **pinfo** info reader is more advanced than the original **info** command. Designed to match the **lynx** text web browser keystrokes, it also adds color. Info nodes for a particular topic are browsed with **pinfo topic**. Enter only **pinfo** for the info topics directory. New documentation nodes become available in **pinfo** when their corresponding software packages are installed.

GNU Info vs. man page navigation

The **info** command uses different navigation keystrokes than does **man**. The **info** command was designed to match the keystrokes of the hypertext-aware **lynx** web browser. Compare the key bindings in the following table:

pinfo and man, key binding comparison

Navigation	pinfo	man
Scroll forward (down) one screen	PageDown or Space	PageDown or Space
Scroll backward (up) one screen	PageUp or b	PageUp or b
Display the directory of topics	d	-
Scroll forward (down) one half-screen	-	d
Display the parent node of a topic	u	-
Display the top (up) of a topic	HOME	1G
Scroll backward (up) one half-screen	-	u
Scroll forward (down) to next hyperlink	DownArrow	-
Open topic at cursor location	Enter	-
Scroll forward (down) one line	-	DownArrow or Enter
Scroll backward (up) to previous hyperlink	UpArrow	-
Scroll backward (up) one line	-	UpArrow
Search for a pattern	/string	/string
Display next node (chapter) in topic	n	-
Repeat previous search forward (down)	/ then Enter	n
Display previous node (chapter) in topic	p	-
Repeat previous search backward (up)	-	N
Quit the program	q	q



References

pinfo info (*Info: An Introduction*)

- All Sections

pinfo pinfo (*Documentation for 'pinfo'*)

- All Sections

The GNU Project

<http://www.gnu.org/gnu/thegnuproject.html>

pinfo(1) and **info(1)** man pages

Practice: Using the pinfo Command

Guided exercise

In this lab, you will browse GNU Info documentation using command-line tools.

Outcomes

Understand program documentation in the GNU Info node system.

Before you begin...

Perform the following steps on serverX unless directed otherwise.

- 1. Invoke **pinfo** without arguments.

```
[student@serverX ~]$ pinfo
```

- 2. Navigate to the **Common options** topic.

Use **UpArrow** or **DownArrow** until **(coreutils) Common options** is highlighted.
Press **Enter** to view this topic.

- 3. Browse through this **info** topic. Learn if long-style options can be abbreviated.

Use **PageUp** and **PageDown** to navigate through the topic. Yes, many programs allow long options to be abbreviated.

- 4. Determine what the symbols **--** signify when used as a command argument.

The symbols **--** signify the end of command *options* and the start of command *arguments* in complex commands where the shell's command-line parser might not correctly make the distinction.

- 5. Without exiting **pinfo**, move up to the **GNU Coreutils** node.

Press **u** to move up to the top node of the topic.

- 6. Move up again, to the top topic.

Press **u** again. Observe that when positioned at the top of a topic node, moving up returns to the directory of topics. Alternately, pressing **d** from any level or topic will move directly to the directory of topics.

- 7. Search for the pattern **nano** and select that topic.

Press **/** followed by the search pattern "nano". With the topic highlighted, press **Enter**.

- 8. In the **Introduction**, locate and select **Command Line Options**. Browse the topic.

Press **Enter** to select **Introduction**, then **DownArrow** and **Enter** to select **Command Line Options**. Use the arrow keys to browse the topic.

- 9. Move up one level to return to **Introduction**. Move to the next topic.

Press **u** to move up one level. The new location will be **nano**'s topic **1 Introduction**. Now press **n**. You will have moved to **nano**'s topic **2 Editor Basics**.

- 10. Exit **pinfo**.

Press **q** to quit **pinfo**.

- 11. Invoke **pinfo** again, specifying **nano** as the destination topic from the command line.

```
[student@serverX ~]$ pinfo nano
```

- 12. Select the **Editor Basics** topic.

Press **DownArrow** to highlight **Editor Basics**, then press **Enter** to select this topic.

- 13. Read the **Entering Text** and **Special Functions** subtopics.

Use arrow keys to highlight a topic, **PageUp** and **PageDown** to browse the text, then press **u** to move up one level. Press **q** to quit **pinfo** when you are finished.

Reading Documentation in /usr/share/doc

Objectives

After completing this section, students should be able to research information using Red Hat Package Manager documentation.

Introducing package documentation

In addition to `man` and `pinfo`, developers may also choose to include documentation in their application's RPM distribution package. When the package is installed, files recognized as documentation are moved to `/usr/share/doc/packagename`. Software package builders may include anything deemed helpful as a complement to, but not duplicating, `man` pages. GNU packages also use `/usr/share/doc` to supplement info nodes.

Most packages include files describing package distribution licensing. Some packages include extensive PDF- or HTML-based documentation. Accordingly, a useful package browsing method is pointing a browser of choice to `file:///usr/share/doc` and utilizing a mouse.

```
[student@desktopX ~]$ firefox file:///usr/share/doc
```

Some packages come with extensive examples, configuration file templates, scripts, tutorials, or user guides. Browse `/usr/share/doc/vsftpd-*` as an example. Some documentation is sparse; the `zip` utility includes the compression algorithm used and little else. Other packages include large user manuals or developer guides, or electronic copies of related, published books.



Note

Developers may choose to bundle extensive documentation in a separate RPM. The `gnuplot` program has the extra `gnuplot-doc` package, which must be installed separately. Other similar packages to browse include `bash-doc` and `samba-doc`. Often, extra packages are found in Red Hat Enterprise Linux's *Optional* software channel.

Many packages also include developer documentation, such as an Application Programming Interface (API) specification, provided in a package with a name ending in `-devel` or similar. Packages may include additional files, such as headers; useful documentation usually only needed for software development or compiling.



Note

The kernel itself has a significant documentation package. The `kernel-doc` package is a treasure of kernel, driver, tuning, and advanced configuration information. Experienced system administrators regularly research `kernel-doc` files.



References

hier(7) man page

- Discusses the hierarchy of Linux directories, including **/usr/share/doc**.

Practice: Viewing Package Documentation

Guided exercise

In this lab, you will research the documentation under `/usr/share/doc` to answer questions. Use your choice of `less`, `gedit`, or a browser to view the documentation file contents.

Outcomes

More familiarity, through practice, with the types of information that developers include with their software packages.

Before you begin...

Perform the following steps on serverX unless directed otherwise.

- 1. Where can you find the latest news about the `vim` project?

```
[student@serverX ~]$ cd /usr/share/doc  
[student@serverX doc]$ less vim-common-*/README.txt
```

View the `vim-common` README and search for "news".

- 2. What is the wiki URL for the `yum` package?

```
[student@serverX doc]$ less yum-3*/README
```

Search for "wiki" in `/usr/share/doc/yum-3*/README`.

- 3. What examples are provided for the command-line `bc` calculator?

```
[student@serverX doc]$ ls -l bc-*/Examples
```

Found in the directory `/usr/share/doc/bc-*/Examples`.

- 4. How would you read the provided GRUB2 manual?

```
[student@serverX doc]$ firefox grub2-tools-*/grub.html
```

Use `firefox` to display `/usr/share/doc/grub2-tools-*/grub.html`.

- 5. What software provides its document as a separate package?

```
[student@serverX doc]$ yum list *-doc*  
[student@serverX doc]$ cd  
[student@serverX ~]$
```

Use `yum` to display only those packages that contain "-doc", "-docs", or "-documentation" in the package name. When finished, return to the home directory.

Getting Help From Red Hat

Objectives

After completing this section, students should be able to view Knowledgebase information and manage support cases from the command line.

Red Hat Customer Portal

Red Hat Customer Portal (<https://access.redhat.com>) provides customers with access to everything provided with their subscription through one convenient location. Customers can search for solutions, FAQs, and articles through Knowledgebase. Access to official product documentation is provided. Support tickets can be submitted and managed. Subscriptions to Red Hat products can be attached to and detached from registered systems, and software downloads, updates, and evaluations can be obtained. Parts of the site are accessible to everyone, while others are exclusive to customers with active subscriptions. Help with getting access to Customer Portal is available at <https://access.redhat.com/help/>.

Customers can work with Red Hat Customer Portal through a web browser. This section will introduce a command line tool that can also be used to access Red Hat Customer Portal services, **redhat-support-tool**.

Knowledgebase



Figure 3.2: Knowledgebase at the Red Hat Customer Portal

Using redhat-support-tool to search Knowledgebase

The Red Hat Support Tool utility **redhat-support-tool** provides a text console interface to the subscription-based Red Hat Access services. Internet access is required to reach the Red Hat Customer Portal. The **redhat-support-tool** is text-based for use from any terminal or SSH connection; no graphical interface is provided.

The **redhat-support-tool** command may be used as an interactive shell or invoked as individually executed commands with options and arguments. The tool's available syntax is identical for both methods. By default, the program launches in shell mode. Use the provided **help** sub-command to see all available commands. Shell mode supports tab completion and the ability to call programs in the parent shell.

```
[student@desktopX ~]$ redhat-support-tool
Welcome to the Red Hat Support Tool.
Command (? for help):
```

When first invoked, **redhat-support-tool** prompts for required Red Hat Access subscriber login information. To avoid repetitively supplying this information, the tool asks to store account information in the user's home directory (`$HOME/.redhat-support-tool/redhat-support-tool.conf`). If a Red Hat Access account is shared by many users, the `--global` option can save account information to

/etc/redhat-support-tool.conf, along with other systemwide configuration. The tool's **config** command modifies tool configuration settings.

The **redhat-support-tool** allows subscribers to search and display the same Knowledgebase content seen on the Red Hat Customer Portal. Knowledgebase permits keyword searches, similar to the man command. Users can enter error codes, syntax from log files, or any mix of keywords to produce a list of relevant solution documents.

The following is an initial configuration and basic search demonstration:

```
[student@desktopX ~]$ redhat-support-tool
Welcome to the Red Hat Support Tool.
Command (? for help): search How to manage system entitlements with subscription-manager
Please enter your RHN user ID: subscriber
Save the user ID in /home/student/.redhat-support-tool/redhat-support-tool.conf (y/n): y
Please enter the password for subscriber: password
Save the password for subscriber in /home/student/.redhat-support-tool/redhat-support-
tool.conf (y/n): y
```

After prompting the user for the required user configuration, the tool continues with the original search request:

```
Type the number of the solution to view or 'e' to return to the previous menu.
1 [ 253273:VER] How to register and subscribe a system to Red Hat Network
(RHN) using Red Hat Subscription Manager (RHSM)?
2 [ 17397:VER] What are Flex Guest Entitlements in Red Hat Network?
3 [ 232863:VER] How to register machines and manage subscriptions using Red
Hat Subscription Manager through an invisible HTTP proxy / Firewall?
3 of 43 solutions displayed. Type 'm' to see more, 'r' to start from the beginning
again, or '?' for help with the codes displayed in the above output.
Select a Solution:
```

Specific sections of solution documents may be selected for viewing.

```
Select a Solution: 1

Type the number of the section to view or 'e' to return to the previous menu.
1 Title
2 Issue
3 Environment
4 Resolution
5 Display all sections
End of options.
Section: 1

Title
=====
How to register and subscribe a system to Red Hat Network (RHN) using Red Hat
Subscription Manager (RHSM)?
URL: https://access.redhat.com/site/solutions/253273
(END) q
[student@desktopX ~]$
```

Directly access Knowledgebase articles by document ID

Locate online articles directly using the tool's **kb** command with the Knowledgebase document ID. Returned documents scroll on the screen without pagination, allowing a user to redirect the output using other local commands. This example views the document with the **less** command:

```
[student@desktopX ~]$ redhat-support-tool kb 253273 | less

Title: How to register and subscribe a system to Red Hat Network (RHN) using Red Hat
Subscription Manager (RHSM)?
ID: 253273
State: Verified: This solution has been verified to work by Red Hat Customers and
Support Engineers for the specified product version(s).
URL: https://access.redhat.com/site/solutions/253273
: q
```

Documents retrieved in unpaginated format are easy to send to a printer, convert to PDF or other document format, or to redirect to a data entry program for an incident tracking or change management system, using other utilities installed and available in Red Hat Enterprise Linux.

Using redhat-support-tool to manage support cases

One benefit of a product subscription is access to technical support through Red Hat Customer Portal. Depending on the system's subscription support level, Red Hat may be contacted through on-line tools or by phone. See https://access.redhat.com/site/support/policy/support_process for links to detailed information about the support process.

Preparing a bug report

Before contacting Red Hat Support, gather relevant information for a bug report.

Define the problem. Be able to clearly state the problem and its symptoms. Be as specific as possible. Detail the steps which will reproduce the problem.

Gather background information. Which product and version is affected? Be ready to provide relevant diagnostic information. This can include output of **sosreport**, discussed later in this section. For kernel problems, this could include the system's **kdump** crash dump or a digital photo of the kernel backtrace displayed on the monitor of a crashed system.

Determine the severity level. Red Hat uses four severity levels to classify issues. *Urgent* and *High* severity problem reports should be followed by a phone call to the relevant local support center (see <https://access.redhat.com/site/support/contact/technicalSupport>).

Severity	Description
<i>Urgent</i> (Severity 1)	A problem that severely impacts your use of the software in a production environment (such as loss of production data or in which your production systems are not functioning). The situation halts your business operations and no procedural workaround exists.
<i>High</i> (Severity 2)	A problem where the software is functioning but your use in a production environment is severely reduced. The situation is causing a high impact to portions of your business operations and no procedural workaround exists.
<i>Medium</i> (Severity 3)	A problem that involves partial, non-critical loss of use of the software in a production environment or development environment. For production environments, there is a medium-to-low impact on your business, but your business continues to function, including by using a procedural workaround. For development environments, where the situation is causing your project to no longer continue or migrate into production.

Severity	Description
Low (Severity 4)	A general usage question, reporting of a documentation error, or recommendation for a future product enhancement or modification. For production environments, there is low-to-no impact on your business or the performance or functionality of your system. For development environments, there is a medium-to-low impact on your business, but your business continues to function, including by using a procedural workaround.

Managing a bug report with **redhat-support-tool**

Subscribers may create, view, modify, and close Red Hat Support cases using **redhat-support-tool**. When support cases are opened or maintained, users may include files or documentation, such as diagnostic reports (sosreport). The tool uploads and attaches files to online cases. Case details including *product*, *version*, *summary*, *description*, *severity*, and *case group* may be assigned with command options or letting the tool prompt for required information. In the following example, the **--product** and **--version** options are specified, but **redhat-support-tool** would provide a list of choices for those options if the **opencase** command did not specify them.

```
[student@desktopX ~]$ redhat-support-tool
Welcome to the Red Hat Support Tool.
Command (? for help): opencase --product="Red Hat Enterprise Linux" --version="7.0"
Please enter a summary (or 'q' to exit): System fails to run without power
Please enter a description (Ctrl-D on an empty line when complete):
When the server is unplugged, the operating system fails to continue.
1 Low
2 Normal
3 High
4 Urgent
Please select a severity (or 'q' to exit): 4
Would you like to assign a case group to this case (y/N)? N
Would see if there is a solution to this problem before opening a support case? (y/N) N
-----
Support case 01034421 has successfully been opened.
```

Including diagnostic information by attaching a SoS report archive

Including diagnostic information when a support case is first created contributes to quicker problem resolution. The **sosreport** command generates a compressed tar archive of diagnostic information gathered from the running system. The **redhat-support-tool** prompts to include one if an archive has been created previously:

```
Please attach a SoS report to support case 01034421. Create a SoS report as
the root user and execute the following command to attach the SoS report
directly to the case:
redhat-support-tool addattachment -c 01034421 path to sosreport

Would you like to attach a file to 01034421 at this time? (y/N) N
Command (? for help):
```

If a current SoS report is not already prepared, an administrator can generate and attach one later, using the tool's **addattachment** command as advised previously. This section's practice exercise will provide the steps for creating and viewing a current SoS diagnostic report.

Support cases can also be viewed, modified, and closed by you as the subscriber:

```
Command (? for help): listcases

Type the number of the case to view or 'e' to return to the previous menu.
1 [Waiting on Red Hat] System fails to run without power
No more cases to display
Select a Case: 1

Type the number of the section to view or 'e' to return to the previous menu.
1 Case Details
2 Modify Case
3 Description
4 Recommendations
5 Get Attachment
6 Add Attachment
7 Add Comment
End of options.
Option: q

Select a Case: q

Command (? for help):q

[student@desktopX ~]$ redhat-support-tool modifycase --status=Closed 01034421
Successfully updated case 01034421
[student@desktopX ~]$
```

The Red Hat Support Tool has advanced application diagnostic and analytic capabilities. Using kernel crash dump core files, **redhat-support-tool** can create and extract a *backtrace*, a report of the active stack frames at the point of a crash dump, to provide onsite diagnostics and open a support case.

The tool also provides log file analysis. Using the tool's **analyze** command, log files of many types, including operating system, JBoss, Python, Tomcat, oVirt, and others, can be parsed to recognize problem symptoms, which can then be viewed and diagnosed individually. Providing preprocessed analysis, as opposed to raw data such as crash dump or log files, allows support cases to be opened and made available to engineers more quickly.



References

sosreport(1) man page

Red Hat Access: Red Hat Support Tool
<https://access.redhat.com/site/articles/445443>

Red Hat Support Tool First Use
<https://access.redhat.com/site/videos/534293>

Contacting Red Hat Technical Support
https://access.redhat.com/site/support/policy/support_process/

Help - Red Hat Customer Portal
<https://access.redhat.com/site/help/>

Practice: Creating and Viewing an SoS Report

Guided exercise

In this lab, you will use the `sosreport` command to generate a SoS report, then view the contents of that diagnostic archive.

Outcomes

A compressed tar archive of systemwide diagnostic information.

Before you begin...

Perform the following steps on serverX unless directed otherwise.

- 1. If currently working as a non-root user, switch to root.

```
[student@serverX ~]$ su -  
Password: redhat
```

- 2. Run the `sosreport` command. This may take many minutes on larger systems.

```
[root@serverX ~]# sosreport  
  
sosreport (version 3.0)  
  
This command will collect system configuration and  
diagnostic information from this Red Hat Enterprise Linux  
system. An archive containing the collected information  
will be generated in /var/tmp and may be provided to a Red  
Hat support representative or used for local diagnostic or  
recording purposes.  
  
Any information provided to Red Hat will be treated in  
strict confidence in accordance with the published support  
policies at:  
  
https://access.redhat.com/support/  
  
The generated archive may contain data considered  
sensitive and its content should be reviewed by the  
originating organization before being passed to any third party.  
  
No changes will be made to system configuration.  
  
Press ENTER to continue, or CTRL-C to quit. ENTER  
  
Please enter your first initial and last name [serverX.example.com]: yourname  
Please enter the case number that you are generating this report for: 01034421
```

Press **Enter**. Provide the requested information. Make up a value for the case number.

```
Running 17/74: general...  
Creating compressed archive...  
  
Your sosreport has been generated and saved in:  
/var/tmp/sosreport-yourname.01034421-20140129000049.tar.xz
```

The checksum is: b2e78125290a4c791162e68da8534887

Please send this file to your support representative.

- 3. Change directory to **/var/tmp**, and unpack the archive.

```
[root@serverX ~]# cd /var/tmp  
[root@serverX tmp]# tar -xvJf sosreport-*.tar.xz
```

- 4. Change directory to the resulting subdirectory and browse the files found there.

```
[root@serverX ~]# cd sosreport-yourname.01034421-20140129000049  
[root@serverX sosreport-yourname.01034421-20140129000049]# ls -lR
```

Open files, list directories, and continue to browse to become familiar with the information included in SoS reports. In the form of the original archived and compressed file, this is the diagnostic information you would be attaching to a **redhat-support-tool** support case. When finished, remove the archive directory and files and return to your home directory.

```
[root@serverX sosreport-yourname.01034421-20140129000049]# cd /var/tmp  
[root@serverX tmp]# rm -rf sosreport*  
[root@serverX tmp]# exit  
[student@serverX ~]$
```

Lab: Viewing and Printing Help Documentation

Performance checklist

In this lab, you will practice research methods typically used by system administrators to learn how to perform necessary tasks.

Outcomes

- Accomplish a given task; practice locating relevant commands by searching **man** pages and **pinfo** nodes.
- Learn new options for commonly used documentation commands.
- Recognize various document file formats; use appropriate tools to view and print documentation and other non-text formatted files.

Before you begin...

Perform the following steps on serverX unless directed otherwise.

1. Research **man(1)** to determine how to prepare a man page for printing. What format or rendering language is commonly used?
2. Create a formatted output file of the **passwd** man page. Determine the file content format.
3. Research using **man** to learn the command(s) used for viewing or printing PostScript files.
4. Research **evince(1)** using **man** to learn how to use the viewer in preview mode. Also, determine how to open a document starting on a specific page.
5. View your PostScript file using the various **evince** options you researched. Close your document file when you are finished.
6. Using **man**, research **lp(1)** to determine how to print any document starting on a specific page. Without actually entering any commands (since there are no printers), what would be the syntax, on one command line, to print only pages 2 and 3 of your PostScript file?

One answer is **lp passwd.ps -P 2-3**.

From **lp(1)**, learn that the **-P** option specifies pages. The **lp** command spools to the *default* printer, sending only the page range starting on 2 and ending on 3.



Note

There are currently no printers configured in the classroom. However, you may practice later using printer models configured in your own environment. Familiarity with these commands is often useful.

7. Using **pinfo**, look for GNU info about the **evince** viewer.
8. As an opportunity to observe the abundance of GNU fundamental utilities, use **pinfo** to locate and browse all document nodes for the *coreutils* commands and programs.

9. Using **firefox**, open the system's package documentation directory and browse into the **man-db** package subdirectory. View the provided manual(s).
10. Using the open **Firefox** browser, locate and browse into the **initscripts** package subdirectory. View the **sysconfig.txt** file, which describes important system configuration options stored in the **/etc/sysconfig** directory.

Solution

In this lab, you will practice research methods typically used by system administrators to learn how to perform necessary tasks.

Outcomes

- Accomplish a given task; practice locating relevant commands by searching **man** pages and **pinfo** nodes.
- Learn new options for commonly used documentation commands.
- Recognize various document file formats; use appropriate tools to view and print documentation and other non-text formatted files.

Before you begin...

Perform the following steps on serverX unless directed otherwise.

- Research **man(1)** to determine how to prepare a man page for printing. What format or rendering language is commonly used?

```
[student@serverX ~]$ man man
```

man uses **-t** to prepare a man page for printing, using PostScript.

- Create a formatted output file of the **passwd** man page. Determine the file content format.

```
[student@serverX ~]$ man -t passwd > passwd.ps  
[student@serverX ~]$ file passwd.ps  
[student@serverX ~]$ less passwd.ps
```

The file is in PostScript format, learned using the **file** command and confirmed by viewing the file contents. Notice the header lines of PostScript information.

- Research using **man** to learn the command(s) used for viewing or printing PostScript files.

```
[student@serverX ~]$ man -k postscript viewer
```

Using multiple words with the **-k** option finds man pages matching *either* word; those with "postscript" or "viewer" in their descriptions. Notice the **evince(1)** and **ghostscript(1)** (or **gs(1)**) commands in the output.

- Research **evince(1)** using **man** to learn how to use the viewer in preview mode. Also, determine how to open a document starting on a specific page.

```
[student@serverX ~]$ man evince
```

The **-w** (or **--preview**) option opens **evince** in preview mode. The **-i** option is used to specify a starting page.

- View your PostScript file using the various **evince** options you researched. Close your document file when you are finished.

```
[student@serverX ~]$ evince passwd.ps
[student@serverX ~]$ evince -w passwd.ps
[student@serverX ~]$ evince -i 3 passwd.ps
```

While normal **evince** mode allows full-screen and presentation-style viewing, the **evince** preview mode is useful for quick browsing and printing. Notice the print icon at the top.

- Using **man**, research **lp(1)** to determine how to print any document starting on a specific page. Without actually entering any commands (since there are no printers), what would be the syntax, on one command line, to print only pages 2 and 3 of your PostScript file?

```
[student@serverX ~]$ man lp
```

One answer is **lp passwd.ps -P 2-3**.

From **lp(1)**, learn that the **-P** option specifies pages. The **lp** command spools to the *default* printer, sending only the page range starting on 2 and ending on 3.



Note

There are currently no printers configured in the classroom. However, you may practice later using printer models configured in your own environment. Familiarity with these commands is often useful.

- Using **pinfo**, look for GNU info about the **evince** viewer.

```
[student@serverX ~]$ pinfo evince
```

Notice that the **evince(1)** man page displays instead. The **pinfo** document viewer looks for the relevant man page when no appropriate GNU documentation node exists for the requested topic. Press **q** to close **pinfo**.

- As an opportunity to observe the abundance of GNU fundamental utilities, use **pinfo** to locate and browse all document nodes for the *coreutils* commands and programs.

```
[student@serverX ~]$ pinfo
```

From the directory node, press **DownArrow** until the link is selected for **Coreutils: Core GNU (file, text, shell) utilities**. Press **Enter** to follow the link to **GNU Coreutils**. Notice the long menu listing, with **Introduction** currently selected. Press **Enter**. At the top of the screen, pay attention to the header, which displays the previous, current, and next nodes. Browse the information, press **n** for the next node, and repeat. Browse each screen, simply noticing the commands and their descriptions. Continue until node **29 Opening the Software Toolbox** is reached. Read this chapter in its entirety using the navigation you have learned. When finished, return the way you came by using **only LeftArrow** until the top directory node is finally reached. Press **q** to close **pinfo**.

9. Using **firefox**, open the system's package documentation directory and browse into the **man-db** package subdirectory. View the provided manual(s).

```
[student@serverX ~]$ firefox /usr/share/doc
```

Remember that bookmarks can be made for any directories that are frequently used. After browsing to the **man-db** directory, click to open and view the text version of the manual, then close it. Click to open the PostScript version. As observed earlier, **evince** is the system's default viewer for PostScript and PDF documents. You may wish to return to these documents later to become more knowledgeable about **man**. When finished, close the **evince** viewer.

10. Using the open **Firefox** browser, locate and browse into the **initscripts** package subdirectory. View the **sysconfig.txt** file, which describes important system configuration options stored in the **/etc/sysconfig** directory.

Notice how useful a browser is for locating and viewing local system documentation. Close the document when finished, but leave Firefox open.

Summary

Reading Documentation Using man Command

An overview of the Linux manual in man page format, including efficient navigation and searching.

Reading Documentation Using pinfo Command

An overview of the GNU Info documentation system, including efficient navigation and searching.

Reading Documentation in /usr/share/doc

The practice of bundling documentation with RPM packages, which are then stored under the directory **/usr/share/doc**.

Getting Help From Red Hat

Use redhat-support-tool to look up Red Hat Knowledgebase articles and manage support cases.



CHAPTER 4

CREATING, VIEWING, AND EDITING TEXT FILES

Overview	
Goal	To create, view, and edit text files from command output or in an editor.
Objectives	<ul style="list-style-type: none">Redirect the text output of a program to a file or to another program.Edit existing text files and create new files from the shell prompt with a text editor.Copy text from a graphical window to a text file using a text editor running in the graphical environment.
Sections	<ul style="list-style-type: none">Redirecting Output to a File or Program (and Practice)Editing Text Files from the Shell Prompt (and Practice)Editing Text Files with a Graphical Editor (and Practice)
Lab	<ul style="list-style-type: none">Creating, Viewing, and Editing Text Files

Redirecting Output to a File or Program

Objectives

After completing this section, students should be able to:

- Describe the technical terms standard input, standard output, and standard error.
- Use redirection characters to control output to files.
- Use piping to control output to other programs.

Standard input, standard output, and standard error

A process structure is constructed with numbered channels (*file descriptors*) to manage open files. Processes connect to files to reach data content or devices these files represent. Processes are created with default connections for channels 0, 1, and 2, known as *standard input*, *standard output*, and *standard error*. Processes use channels 3 and above to connect to other files.

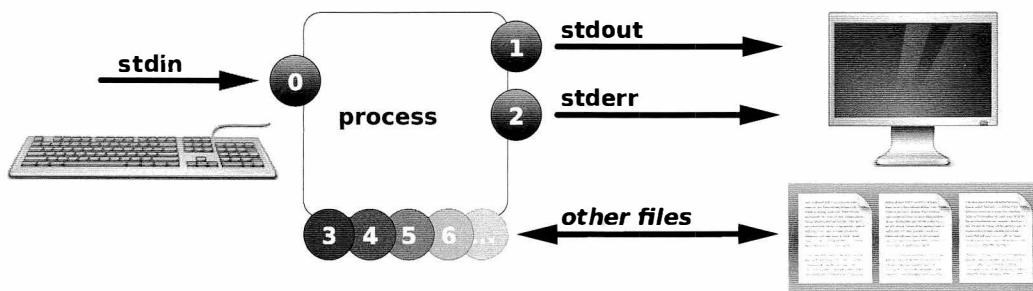


Figure 4.1: Process I/O channels (file descriptors)

Channels (File Descriptors)

Number	Channel name	Description	Default connection	Usage
0	stdin	Standard input	Keyboard	read only
1	stdout	Standard output	Terminal	write only
2	stderr	Standard error	Terminal	write only
3+	filename	Other files	none	read and/or write

Redirecting output to a file

Channel *redirection* replaces default channel destinations with file names representing either output files or devices. Using redirection, process output and error messages can be captured as file contents, sent to a device, or discarded.

Redirecting **stdout** suppresses process output from appearing on the terminal. As seen in the following table, redirecting *only* **stdout** does not suppress **stderr** error messages from displaying on the terminal. The special file **/dev/null** quietly discards channel output redirected to it.

Output Redirection Operators

Usage	Explanation (note)	Visual aid
<code>>file</code>	redirect stdout to a file ⁽¹⁾	
<code>>>file</code>	redirect stdout to a file, append to current file content ⁽²⁾	
<code>2>file</code>	redirect stderr to a file ⁽¹⁾	
<code>2>/dev/null</code>	discard stderr error messages by redirecting to /dev/null	
<code>&>file</code>	combine stdout and stderr to one file ⁽¹⁾	
<code>>>file 2>&1</code>	combine stdout and stderr , append to current file content ⁽²⁾⁽³⁾	
Note:	<p>⁽¹⁾Overwrite existing file, create file if new. ⁽²⁾Append existing file, create file if new. ⁽³⁾The order of redirection is important to avoid unexpected command behavior. 2>&1 sends stderr to the same place as stdout. For this to work, stdout needs to be redirected first, before adding stderr to stdout. Although &>> is an alternate way to append both stdout and stderr to a file, 2>&1 is the method needed to send both stdout and stderr through a pipe.</p>	

Examples for output redirection

Many routine administration tasks are simplified by using redirection. Use the previous table to assist while considering the following examples:

- Save a timestamp for later reference.

```
[student@desktopX ~]$ date > /tmp/saved-timestamp
```

- Copy the last 100 lines from a log file to another file.

```
[student@desktopX ~]$ tail -n 100 /var/log/dmesg > /tmp/last-100-boot-messages
```

- Concatenate four files into one.

```
[student@desktopX ~]$ cat file1 file2 file3 file4 > /tmp/all-four-in-one
```

- List the home directory's hidden and regular file names into a file.

```
[student@desktopX ~]$ ls -a > /tmp/my-file-names
```

- Append output to an existing file.

```
[student@desktopX ~]$ echo "new line of information" >> /tmp/many-lines-of-information  
[student@desktopX ~]$ diff previous-file current-file >> /tmp/tracking-changes-made
```

- In the next examples, errors are generated since normal users are denied access to system directories. Redirect errors to a file while viewing normal command output on the terminal.

```
[student@desktopX ~]$ find /etc -name passwd 2> /tmp/errors
```

- Save process output and error messages to separate files.

```
[student@desktopX ~]$ find /etc -name passwd > /tmp/output 2> /tmp/errors
```

- Ignore and discard error messages.

```
[student@desktopX ~]$ find /etc -name passwd > /tmp/output 2> /dev/null
```

- Store output and generated errors together.

```
[student@desktopX ~]$ find /etc -name passwd &> /tmp/save-both
```

- Append output and generated errors to an existing file.

```
[student@desktopX ~]$ find /etc -name passwd >> /tmp/save-both 2>&1
```

Constructing pipelines

Redirection controls channel output to or from *files* while *piping* sends channel output to another process.

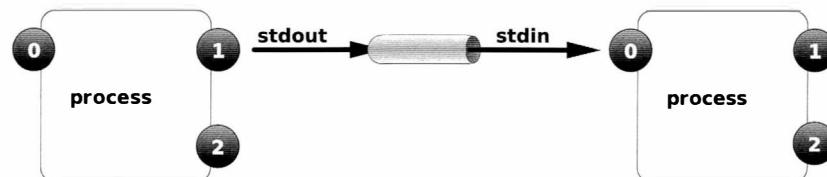


Figure 4.8: Process I/O piping

Examples for process piping redirection

Paginate a command's long output.

```
[student@desktopX ~]$ ls -l /usr/bin | less
```

Count the number of lines in an output or listing.

```
[student@desktopX ~]$ ls | wc -l > /tmp/how-many-files
```

Grab the first lines, last lines, or selected lines of command output.

```
[student@desktopX ~]$ ls -t | head -n 10 > /tmp/ten-last-changed-files
```

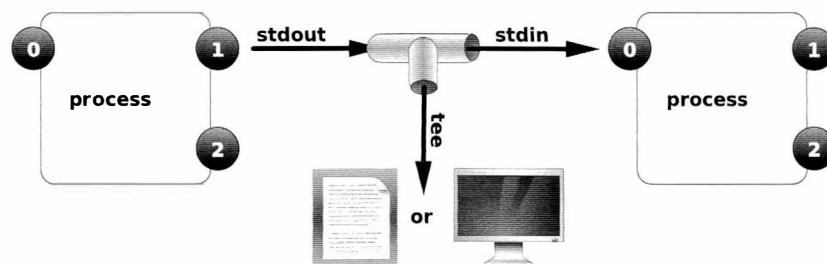


Figure 4.9: Process I/O piping with tee

Examples for using the tee command for piping

The **tee** command displays or redirects the intermediate result normally suppressed due to piping. In the first example, the **ls** listing is viewed on a terminal while simultaneously being stored in a file.

```
[student@desktopX ~]$ ls -l | tee /tmp/saved-output
```

Determine the terminal device for the current window. Send the results as mail and view the same results in this window.

```
[student@desktopX ~]$ tty
/dev/pts/0
```

```
[student@desktopX ~]$ ls -l | tee /dev/pts/0 | mail -s subject  
student@desktop1.example.com
```



References

info bash (*The GNU Bash Reference Manual*)

- Section 3.2.2: Pipelines

info bash (*The GNU Bash Reference Manual*)

- Section 3.6: Redirections

info coreutils 'tee invocation' (*The GNU coreutils Manual*)

- Section 17.1: Redirect output to multiple files or processes

bash(1), **cat(1)**, **head(1)**, **less(1)**, **mail(1)**, **tee(1)**, **tty(1)**, **wc(1)** man pages

Practice: I/O Redirection and Pipelines

Quiz

Match the following items to their counterparts in the table.

&>/dev/null	&>file	2>/dev/null	> file 2> /dev/null
>>file 2>&1	>file 2>file2	tee file	

Result needed	Redirection syntax used
Display command output to terminal, ignore all errors.	
Send command output to file; errors to different file.	
Send output and errors to the same new, empty file.	
Send output and errors to the same file, but preserve existing file content.	
Run a command, but throw away all possible terminal displays.	
Send command output to both the screen and a file at the same time.	
Run command, save output in a file, discard error messages.	

Solution

Match the following items to their counterparts in the table.

Result needed	Redirection syntax used
Display command output to terminal, ignore all errors.	2>/dev/null
Send command output to file; errors to different file.	>file 2>file2
Send output and errors to the same new, empty file.	&>file
Send output and errors to the same file, but preserve existing file content.	>>file 2>&1
Run a command, but throw away all possible terminal displays.	&>/dev/null
Send command output to both the screen and a file at the same time.	tee file
Run command, save output in a file, discard error messages.	> file 2> /dev/null

Editing Text Files from the Shell Prompt

Objectives

After completing this section, students should be able to:

- Create new files and edit existing text files from the shell prompt.
- Navigate within an editor to effectively accomplish editing tasks.

Editing files with Vim

A key design principle of Linux is that information is stored in text-based files. Text files include both *flat files* with rows of similar information, such as configuration files in `/etc`, and *Extensible Markup Language (XML)* files, which define data structure through text tags, seen in application configuration files throughout both `/etc` and `/usr`. The advantage of text files is that they can be moved or shared between systems without requiring conversion, and can be viewed and edited using any simple text editor.

Vim is an improved version of the vi editor distributed with Linux and UNIX systems. Vim is highly configurable and efficient for practiced users, including such features as split screen editing, color formatting, and highlighting for editing text.

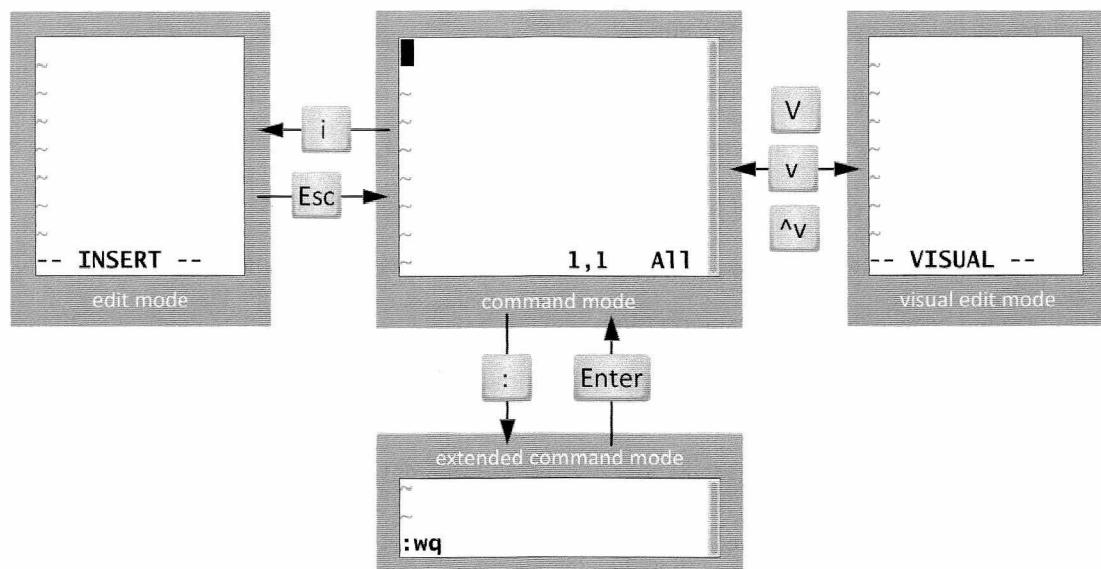


Figure 4.10: Moving between Vim modes

When first opened, Vim starts in *command mode*, used for navigation, cut and paste, and other text manipulation. Enter each of the other modes with single character keystrokes to access specific editing functionality:

- An **i** keystroke enters *insert mode*, where all text typed becomes file content. Pressing **Esc** returns to *command mode*.
- A **v** keystroke enters *visual mode*, where multiple characters may be selected for text manipulation. Use **V** for multi-line and **Ctrl-v** for block selection. The same keystroke used to enter *visual mode* (**v**, **V** or **Ctrl-v**) is used to exit.

- The `:` keystroke begins *extended command mode* for tasks like writing the file to save it, and quitting the Vim editor.

The minimum, basic Vim workflow

Vim has efficient, coordinated keystrokes for advanced editing tasks. Although considered useful with practice, Vim's capabilities can overwhelm new users. The following workflow presents the *minimum* keystrokes every Vim user must learn to accomplish *any* editing task.

The instructor will demonstrate a typical file editing session using only basic Vim keystrokes.

1. Open a file with **vim filename**.
2. Repeat this text entry cycle, as many times as the task requires:
 - Use arrow keys to position the cursor.
 - Press **i** to enter insert mode.
 - Enter text.
 - Press **Esc** to return to command mode.
 - If necessary, press **u** to undo mistaken edits on the current line.
3. Repeat this text deletion cycle, as many times as the task requires:
 - Use arrow keys to position the cursor.
 - Press **x** to delete a selection of text.
 - If necessary, use **u** to undo mistaken edits on the current line.
4. To save or exit, choose one of the following to write or discard file edits:
 - Enter **:w** to write (save) the file and remain in command mode for more editing.
 - Enter **:wq** to write the file and quit Vim.
 - Enter **:q!** to quit Vim, but discard all file changes since the last write.

Rearranging existing text

In Vim, copy and paste is known as *yank and put*, using command characters **y** and **p**. Begin by positioning the cursor on the first character to be selected, then enter visual mode. Use arrow keys to expand the visual selection. When ready, press **y** to *yank* the selection into memory. Position the cursor at the new location, then press **p** to *put* the selection at the cursor.

The instructor will demonstrate “yank and put” using visual mode.

1. Open a file with **vim filename**.
2. Repeat this text selection cycle, as many times as the task requires:
 - Use arrow keys to position the cursor to the first character.
 - Press **v** to enter visual mode.
 - Use arrow keys to position the cursor to the last character.

- Press **y** to yank (copy) the selection.
 - Use arrow keys to position the cursor at the insert location.
 - Press **p** to put (paste) the selection.
3. To save or exit, choose one of the following to write or discard file edits:
- Enter **:w** to write (save) the file and remain in command mode for more editing.
 - Enter **:wq** to write the file and quit Vim.
 - Enter **:q!** to quit Vim, but discard all file changes since the last write.



Note

Beware the advanced Vim user offering shortcuts and tricks before the basics are mastered. Vim requires practice to become efficient. It is recommended to continue learning new keystrokes to extend Vim's usefulness. For those curious how extensive this can be, perform an Internet search for "Vim tips".

An in-depth presentation of **Vim** is included in the *Red Hat Enterprise Linux 7 System Administration II* course.



References

vim(1) man page

Vim the editor
<http://www.vim.org/>

Practice: Editing Files with Vim

Guided exercise

In this lab, you will use a locally installed resource to practice entry-level **vim** editor techniques.

Outcomes:

Experience with vim, and knowledge about using **vimtutor** to gain competency.

Before you begin...

In this exercise, use the existing Vim tutorial bundled with the Vim editor. The installed *vim-enhanced* package provides **vimtutor**. For each exercise step, use the corresponding lesson in vimtutor to practice. Return here when the lesson step is complete. Perform the following steps on serverX unless directed otherwise.

- 1. Open **vimtutor**. Read the Welcome screen and perform *Lesson 1.1*.

```
[student@serverX ~]$ vimtutor
```

In the lecture, only keyboard arrow keys were used for navigation. In **vi**'s early years, users could not rely on working keyboard mappings for arrow keys. Therefore, **vi** was designed with commands using only standard character keys, such as the conveniently grouped **h**, **j**, **k**, and **l**. Here is one way to remember them:

hang back, jump down, kick up, leap forward.

- 2. Return to the **vimtutor** window. Perform *Lesson 1.2*.

This early lesson teaches how to quit without having to keep an unwanted file change. All changes are lost, but this is better than leaving a critical file in an incorrect state.

- 3. Return to the **vimtutor** window. Perform *Lesson 1.3*.

Vim has faster, more efficient keystrokes to delete an exact amount of words, lines, sentences, and paragraphs. However, any editing job *can* be accomplished using only **x** for single-character deletion.

- 4. Return to the **vimtutor** window. Perform *Lesson 1.4*.

The minimum required keystrokes are for entering and leaving edit mode, arrow keys, and deleting. For most edit tasks, the first key pressed is **i**.

- 5. (Optional) Return to the **vimtutor** window. Perform *Lesson 1.5*.

In the lecture, only the **i** (*insert*) command was taught as the keystroke to enter edit mode. This vimtutor lesson demonstrates that other keystrokes are available to change the cursor placement when insert mode is entered. However, once in insert mode, all text typed is still file content.

- 6. Return to the **vimtutor** window. Perform *Lesson 1.6*.

Save the file by writing and quitting. This is the last lesson for the minimum *required* keystrokes to be able to accomplish any editing task.

- 7. Return to the **vimtutor** window. Finish by reading the *Lesson 1 Summary*.

There are six more multi-step lessons in **vimtutor**. None are assigned as further lessons for this course, but feel free to use **vimtutor** on your own to learn more about **Vim**.

Editing Text Files with a Graphical Editor

Objectives

After completing this section, students should be able to:

- Edit text files with gedit.
- Copy text between graphical windows.

Editing files with gedit

The **gedit** application is a full-featured text editor for the GNOME desktop environment. Launch **gedit** by selecting **Applications > Accessories > gedit** from the GNOME menu. Like other graphical applications, **gedit** can be started without navigating the menu. Press **Alt+F2** to open the **Enter a Command** dialog box. Type **gedit** and press **Enter**.

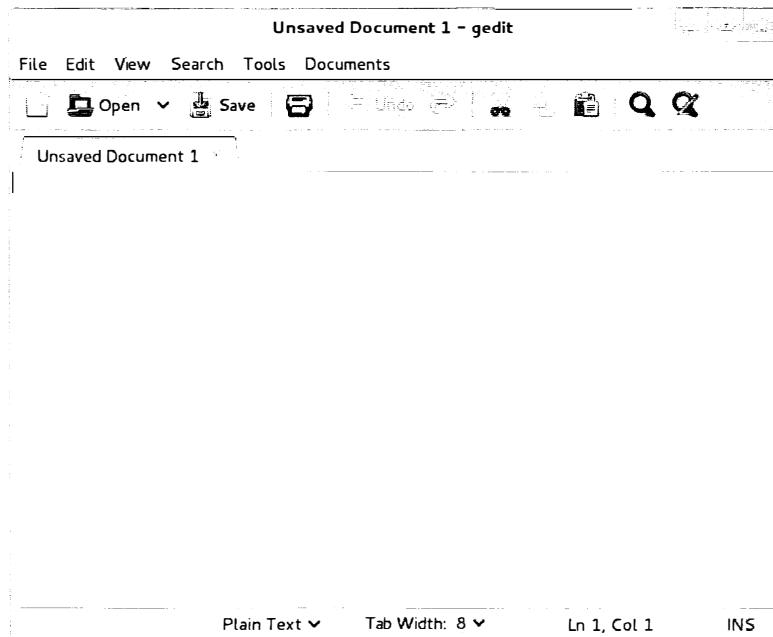


Figure 4.11: gedit text editor

GNOME Help includes a **gedit** help guide, which may be found by selecting **Applications > Favorites > Help** from the GNOME menu. Then select **Go > All Documents** to view the list of graphical applications. Scroll down to select the **gedit Text Editor** hyperlink.

Basic gedit Keystrokes

Perform many file management tasks using **gedit**'s menu:

- To create a new file in **gedit**, click the blank paper toolbar icon, or select **File > New (Ctrl-n)** from the menu.
- To save a file, click the disk-drive save toolbar icon, or select **File > Save (Ctrl-s)** from the menu.

- To open an existing file, click the Open toolbar icon, or select File > Open (**Ctrl-o**) from the menu. The Open Files dialog window will display from which users can locate and select the file to open.

Multiple files may be opened simultaneously, each with a unique tab under the menu bar. Tabs display a file name after being saved the first time.

Copying text between graphical windows

Text can be copied between documents, text windows, and command windows in the graphical environment. Selected text is duplicated using *copy and paste* or moved using *cut and paste*. Whether cut or copied, the text is held in memory for pasting into another location.

To select text:

- Click and hold the left mouse button before the first character desired.
- Drag the mouse over and down until all required text is in a single highlighted selection, then release the left button. Do not click again with the left button, as that deselects the text.

To paste the selection, multiple methods can accomplish the same result. In the first method:

- Click the right mouse button anywhere on the text area just selected.
- From the resulting context menu, select *either cut or copy*.
- Move the mouse to the window or document where the text is to be placed, click the left mouse button to position where the text should go, and click the right mouse button again, now choosing **paste**.

Here is a shorter mouse technique to practice:

- First, select the text.
- Hover the mouse over the destination window and click the center mouse button, just once, to paste the text at the cursor.

This last method can only copy, not cut. The original text remains selected and can be deleted. As with other methods, the text remains in memory and can be repeatedly pasted.

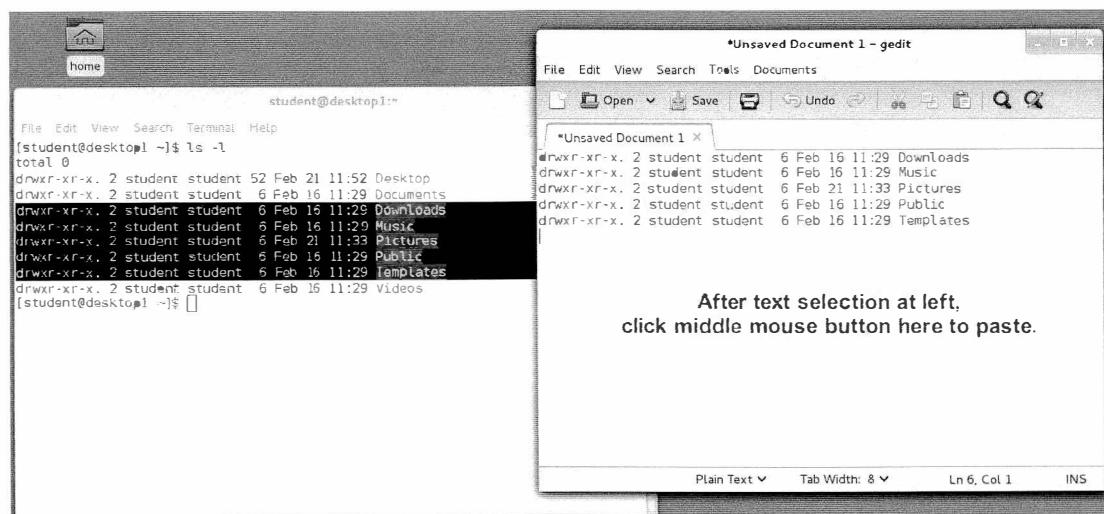


Figure 4.12: Select and paste using middle mouse button

The keyboard shortcut method can also be used in graphical applications:

- First, select the text.
- Use **Ctrl-x** to cut or **Ctrl-c** to copy the text.
- Click the location where the text is to be placed to position the cursor.
- Use **Ctrl-v** to paste.



Important

Ctrl-c and **Ctrl-v** will not copy and paste within a terminal window. **Ctrl-c** will actually terminate the current running process within a terminal window. To copy and paste within a terminal window, use **Ctrl-shift-c** and **Ctrl-shift-v**.



References

gedit(1) man page

gedit Text Editor

- **yelp help:gedit**

gedit Wiki

<https://wiki.gnome.org/Apps/Gedit>

Practice: Copying Text Between Windows

Guided exercise

In this lab, you will edit a file with gedit, selecting text and pasting it into the editor.

Outcomes:

An edited list of the configuration files found in the user's home directory.

Before you begin...

Perform the following steps on serverX unless directed otherwise. Log in as **student** and begin in student's home directory.

- 1. Redirect a long listing of all home directory files, including hidden, into a file named **gedit_lab.txt**. Confirm that the file contains the listing.

```
[student@serverX ~]$ cd  
[student@serverX ~]$ ls -al > gedit_lab.txt  
[student@serverX ~]$ cat gedit_lab.txt
```

- 2. Open the file with the **gedit** text editor. Include the ending ampersand so that the shell prompt can return while **gedit** is running.

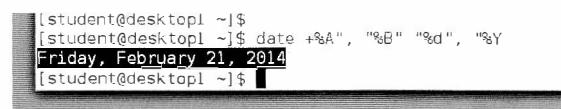
```
[student@serverX ~]$ gedit gedit_lab.txt &
```

- 3. Insert the date at the top of your file document.

- 3.1. In the shell command window, display today's date with day of the week, month, date, and year.

```
[student@serverX ~]$ date +%A", "%B" "%d", "%Y  
Friday, February 21, 2014
```

- 3.2. Select the text using the mouse.



```
[student@desktop1 ~]$  
[student@desktop1 ~]$ date +%A", "%B" "%d", "%Y  
Friday, February 21, 2014  
[student@desktop1 ~]$
```

- 3.3. Insert the text at the top of the file document. Switch to the **gedit** window. Using arrow keys, place the cursor at the upper-left corner of the document. Press the middle mouse button to paste the text.

- 3.4. Press **Enter** one or more times at the end of the inserted text to open blank lines above the file listing.

- 4. Insert a description for this document, including your username and host name, on line 2.

- 4.1. In the shell command window, create descriptive text using shell expansion concepts to include the username and host name where the file list was generated.

```
[student@serverX ~]$ echo "$USER's configuration files on" $(hostname)  
student's configuration files on serverX.example.com
```

- 4.2. Select the text using the mouse.

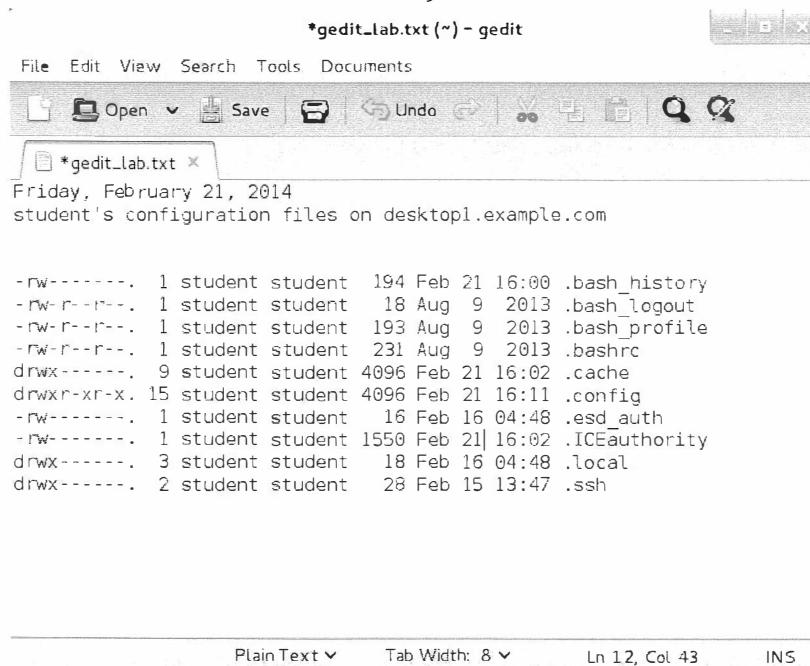
```
[student@desktop1 ~]$  
[student@desktop1 ~]$ echo "$USER's configuration files on" $(hostname)  
student's configuration files on desktop1.example.com  
[student@desktop1 ~]$
```

- 4.3. Insert the text on the second line of the file document. Switch to the **gedit** window. Using the arrow keys, place the cursor at the second line's leftmost character. Press the middle mouse button to paste the text.
 - 4.4. Press **Enter** or **Delete**, as necessary, to maintain blank lines above the file listing.

□ 5. Remove file lines that are not hidden configuration files or directories.

 - 5.1. Remove the "total" line at the beginning of the listing.
 - 5.2. Remove the two lines representing the current directory and the parent directory.
 - 5.3. Remove lines for file names that do *not* start with a dot. Do not edit or remove lines for hidden files or directories that begin with a dot.

□ 6. The final file document should be similar to the following image. Manually edit the file to make corrections. The asterisk in the document tab or window header is a reminder of edits unsaved. Save the file and exit gedit.



Lab: Creating, Viewing, and Editing Text Files

Performance checklist

In this lab, you will edit a file using Vim's visual mode to simplify repetitive edits.

Outcomes:

Familiarity with the utilities and techniques required to perform file editing. The final edited file will be a list of selected files and tabular data.

Before you begin...

Perform the following steps on serverX unless directed otherwise. Log in as **student** and begin in the student's home directory.

1. Redirect a long listing of all content in student's home directory, including hidden directories and files, into a file named **editing_final_lab.txt**. Your home directory files may not exactly match those shown in the example graphics. This lab edits arbitrary lines and columns. The important outcome is to practice the visual selection process.
2. Edit the file using Vim, to take advantage of *visual mode*.
3. Remove the first three lines, since those lines are not normal file names. Enter line-based visual mode with upper case **V**.
4. Remove the permission columns for group and world on the first line. In this step, enter visual mode with lower case **v**, which allows selecting characters on a single line only.
5. Remove the permission columns for group and world on the remaining lines. This step will use a more efficient block selection visual mode to avoid having to repeat the single line edit multiple times. This time, enter visual mode with the control sequence **Ctrl-v**, which allows selecting a block of characters on multiple lines.
6. Remove the *group owner* column, leaving only one "student" column on all lines. Use the same block selection technique as the last step.
7. Remove the time column, but leave the month and day on all lines. Again, use the block selection visual mode.
8. Remove the **Desktop** and **Public** rows. This time, enter visual mode with upper case **V**, which automatically selects full lines.
9. Save and exit. Make a backup, using the date (in seconds) to create a unique file name.
10. Mail the file contents as the message, not an attachment, to the user **student**.
11. Append a dashed line to the file to recognize the beginning of newer content.
12. Append a full process listing, but only for processes owned by the current user **student** and running on the currently used terminal. View the process listing and send the listing to the file with one command line.

13. Confirm that the process listing is at the bottom of the lab file.

Solution

In this lab, you will edit a file using Vim's visual mode to simplify repetitive edits.

Outcomes:

Familiarity with the utilities and techniques required to perform file editing. The final edited file will be a list of selected files and tabular data.

Before you begin...

Perform the following steps on serverX unless directed otherwise. Log in as **student** and begin in the student's home directory.

1. Redirect a long listing of all content in student's home directory, including hidden directories and files, into a file named **editing_final_lab.txt**. Your home directory files may not exactly match those shown in the example graphics. This lab edits arbitrary lines and columns. The important outcome is to practice the visual selection process.

```
[student@serverX ~]$ cd  
[student@serverX ~]$ ls -al > editing_final_lab.txt
```

2. Edit the file using Vim, to take advantage of *visual mode*.

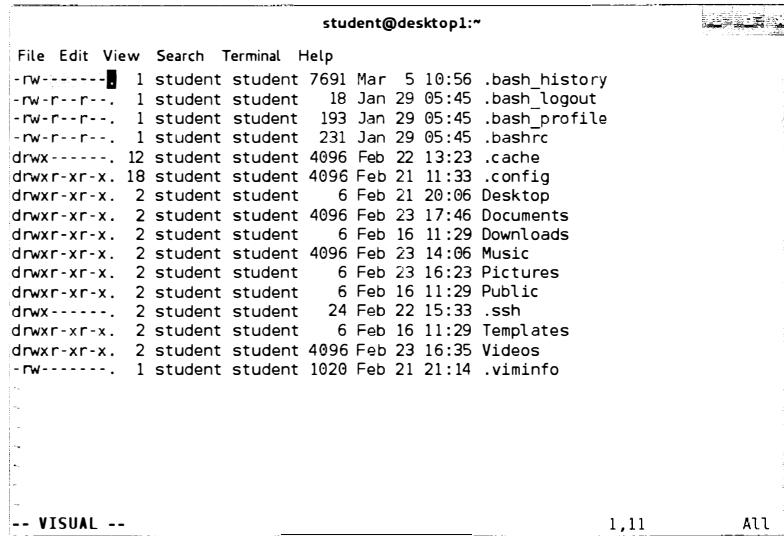
```
[student@serverX ~]$ vim editing_final_lab.txt
```

3. Remove the first three lines, since those lines are not normal file names. Enter line-based visual mode with upper case **V**.

Use the arrow keys to position the cursor at the first character in the first row. Enter line-based visual mode with **V**. Move down using the down arrow key twice to select the first three rows. Delete the rows with **x**.

4. Remove the permission columns for group and world on the first line. In this step, enter visual mode with lower case **v**, which allows selecting characters on a single line only.

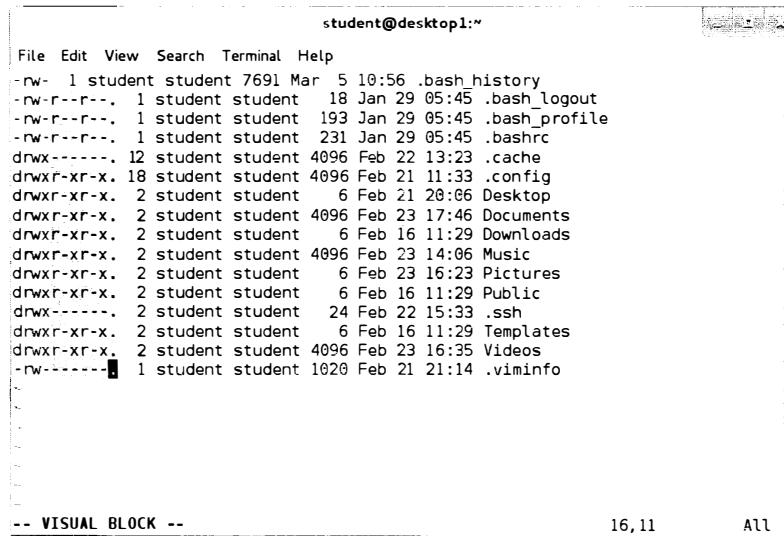
Use the arrow keys to position the cursor at the first character. Enter visual mode with **v**. Use the arrow keys to position the cursor at the last character, as shown in the screenshot. Delete the selection with **x**.



```
student@desktop1:~
File Edit View Search Terminal Help
-rw----- 1 student student 7691 Mar  5 10:56 .bash_history
-rw-r--r--. 1 student student 18 Jan 29 05:45 .bash_logout
-rw-r--r--. 1 student student 193 Jan 29 05:45 .bash_profile
-rw-r--r--. 1 student student 231 Jan 29 05:45 .bashrc
drwx----- 12 student student 4096 Feb 22 13:23 .cache
drwxr-xr-x. 18 student student 4096 Feb 21 11:33 .config
drwxr-xr-x. 2 student student 6 Feb 21 20:06 Desktop
drwxr-xr-x. 2 student student 4096 Feb 23 17:46 Documents
drwxr-xr-x. 2 student student 6 Feb 16 11:29 Downloads
drwxr-xr-x. 2 student student 4096 Feb 23 14:06 Music
drwxr-xr-x. 2 student student 6 Feb 23 16:23 Pictures
drwxr-xr-x. 2 student student 6 Feb 16 11:29 Public
drwx----- 2 student student 24 Feb 22 15:33 .ssh
drwxr-xr-x. 2 student student 6 Feb 16 11:29 Templates
drwxr-xr-x. 2 student student 4096 Feb 23 16:35 Videos
-rw----- 1 student student 1020 Feb 21 21:14 .viminfo
```

- Remove the permission columns for group and world on the remaining lines. This step will use a more efficient block selection visual mode to avoid having to repeat the single line edit multiple times. This time, enter visual mode with the control sequence **Ctrl-v**, which allows selecting a block of characters on multiple lines.

Use the arrow keys to position the cursor at the first character. Enter visual mode with the control sequence **Ctrl-v**. Use the arrow keys to position the cursor at the last character of the column on the last line, as shown in the screenshot. Delete the selection with **x**.



```
student@desktop1:~
File Edit View Search Terminal Help
-rw- 1 student student 7691 Mar  5 10:56 .bash_history
-rw-r--r--. 1 student student 18 Jan 29 05:45 .bash_logout
-rw-r--r--. 1 student student 193 Jan 29 05:45 .bash_profile
-rw-r--r--. 1 student student 231 Jan 29 05:45 .bashrc
drwx----- 12 student student 4096 Feb 22 13:23 .cache
drwxr-xr-x. 18 student student 4096 Feb 21 11:33 .config
drwxr-xr-x. 2 student student 6 Feb 21 20:06 Desktop
drwxr-xr-x. 2 student student 4096 Feb 23 17:46 Documents
drwxr-xr-x. 2 student student 6 Feb 16 11:29 Downloads
drwxr-xr-x. 2 student student 4096 Feb 23 14:06 Music
drwxr-xr-x. 2 student student 6 Feb 23 16:23 Pictures
drwxr-xr-x. 2 student student 6 Feb 16 11:29 Public
drwx----- 2 student student 24 Feb 22 15:33 .ssh
drwxr-xr-x. 2 student student 6 Feb 16 11:29 Templates
drwxr-xr-x. 2 student student 4096 Feb 23 16:35 Videos
-rw----- 1 student student 1020 Feb 21 21:14 .viminfo
```

- Remove the *group owner* column, leaving only one "student" column on all lines. Use the same block selection technique as the last step.

Use the arrow keys to position the cursor at the first character of the group owner column. Enter visual mode with **Ctrl-v**. Use the arrow keys to position the cursor at the last character and row of the group owner column, as shown in the screenshot. Delete the selection with **x**.

```
student@desktop1:~
```

```

File Edit View Search Terminal Help
-rw- 1 student student 7691 Mar 5 10:56 .bash_history
-rw- 1 student student 18 Jan 29 05:45 .bash_logout
-rw- 1 student student 193 Jan 29 05:45 .bash_profile
-rwx 1 student student 231 Jan 29 05:45 .bashrc
drwx 12 student student 4096 Feb 22 13:23 .cache
drwx 18 student student 4096 Feb 21 11:33 .config
drwx 2 student student 6 Feb 21 20:06 Desktop
drwx 2 student student 4096 Feb 23 17:46 Documents
drwx 2 student student 6 Feb 16 11:29 Downloads
drwx 2 student student 4096 Feb 23 14:06 Music
drwx 2 student student 6 Feb 23 16:23 Pictures
drwx 2 student student 6 Feb 16 11:29 Public
drwx 2 student student 24 Feb 22 15:33 .ssh
drwx 2 student student 6 Feb 16 11:29 Templates
drwx 2 student student 4096 Feb 23 16:35 Videos
-rw- 1 student student 1020 Feb 21 21:14 .viminfo

```

-- VISUAL_BLOCK -- 16,23 All

- Remove the time column, but leave the month and day on all lines. Again, use the block selection visual mode.

Use the arrow keys to position the cursor at the first character. Enter visual mode with **Ctrl-v**. Use the arrow keys to position the cursor at the last character and row of the time column, as shown in the screenshot. Delete the selection with **x**.

```
student@desktop1:~
```

```

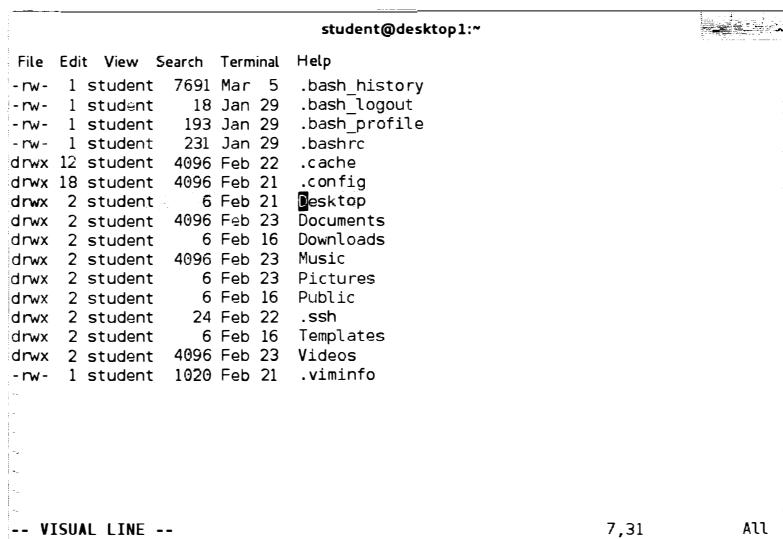
File Edit View Search Terminal Help
-rw- 1 student student 7691 Mar 5 10:56 .bash_history
-rw- 1 student student 18 Jan 29 05:45 .bash_logout
-rw- 1 student student 193 Jan 29 05:45 .bash_profile
-rwx 1 student student 231 Jan 29 05:45 .bashrc
drwx 12 student student 4096 Feb 22 13:23 .cache
drwx 18 student student 4096 Feb 21 11:33 .config
drwx 2 student student 6 Feb 21 20:06 Desktop
drwx 2 student student 4096 Feb 23 17:46 Documents
drwx 2 student student 6 Feb 16 11:29 Downloads
drwx 2 student student 4096 Feb 23 14:06 Music
drwx 2 student student 6 Feb 23 16:23 Pictures
drwx 2 student student 6 Feb 16 11:29 Public
drwx 2 student student 24 Feb 22 15:33 .ssh
drwx 2 student student 6 Feb 16 11:29 Templates
drwx 2 student student 4096 Feb 23 16:35 Videos
-rw- 1 student student 1020 Feb 21 21:14 .viminfo

```

-- VISUAL_BLOCK -- 16,34 All

- Remove the **Desktop** and **Public** rows. This time, enter visual mode with upper case **V**, which automatically selects full lines.

Use the arrow keys to position the cursor at any character on the **Desktop** row. Enter visual mode with upper case **V**. The full line is selected, as shown in the screenshot. Delete the selection with **x**. Repeat for the **Public** row.



A screenshot of a terminal window titled "student@desktop1:~". The window displays a file listing in the current directory (~). The files listed include .bash_history, .bash_logout, .bash_profile, .bashrc, .cache, .config, .Desktop, .Documents, .Downloads, .Music, .Pictures, .Public, .ssh, .Templates, .Videos, and .viminfo. The listing shows permissions (e.g., -rw-, drwx), file size (e.g., 7691, 4096), modification date (e.g., Mar 5, Feb 22), and file names.

```
student@desktop1:~  
File Edit View Search Terminal Help  
-rw- 1 student 7691 Mar 5 .bash_history  
-rw- 1 student 18 Jan 29 .bash_logout  
-rw- 1 student 193 Jan 29 .bash_profile  
-rw- 1 student 231 Jan 29 .bashrc  
drwx 12 student 4096 Feb 22 .cache  
drwx 18 student 4096 Feb 21 .config  
drwx 2 student 4096 Feb 21 .Desktop  
drwx 2 student 4096 Feb 23 .Documents  
drwx 2 student 6 Feb 16 .Downloads  
drwx 2 student 4096 Feb 23 .Music  
drwx 2 student 6 Feb 23 .Pictures  
drwx 2 student 6 Feb 16 .Public  
drwx 2 student 24 Feb 22 .ssh  
drwx 2 student 6 Feb 16 .Templates  
drwx 2 student 4096 Feb 23 .Videos  
-rw- 1 student 1020 Feb 21 .viminfo  
  
-- VISUAL LINE -- 7,31 All
```

9. Save and exit. Make a backup, using the date (in seconds) to create a unique file name.

```
[student@serverX ~]$ cp editing_final_lab.txt editing_final_lab_$(date +%s).txt
```

10. Mail the file contents as the message, not an attachment, to the user student.

```
[student@serverX ~]$ cat editing_final_lab.txt | mail -s "lab file" student
```

11. Append a dashed line to the file to recognize the beginning of newer content.

```
[student@serverX ~]$ echo "-----" >> editing_final_lab.txt
```

12. Append a full process listing, but only for processes owned by the current user **student** and running on the currently used terminal. View the process listing and send the listing to the file with one command line.

```
[student@serverX ~]$ ps -f | tee -a editing_final_lab.txt
```

13. Confirm that the process listing is at the bottom of the lab file.

```
[student@serverX ~]$ cat editing_final_lab.txt  
-rw- 1 student 7691 Mar 5 .bash_history  
-rw- 1 student 18 Jan 29 .bash_logout  
-rw- 1 student 193 Jan 29 .bash_profile  
-rw- 1 student 231 Jan 29 .bashrc  
drwx 12 student 4096 Feb 22 .cache  
drwx 18 student 4096 Feb 21 .config  
drwx 2 student 4096 Feb 23 .Documents  
drwx 2 student 6 Feb 16 .Downloads  
drwx 2 student 4096 Feb 23 .Music  
drwx 2 student 6 Feb 23 .Pictures  
drwx 2 student 24 Feb 22 .ssh  
drwx 2 student 6 Feb 16 .Templates  
drwx 2 student 4096 Feb 23 .Videos
```

```
-rw- 1 student 1020 Feb 21 .viminfo

-----
UID      PID  PPID  C STIME TTY          TIME CMD
student  2005  2001  0 16:01 pts/0    00:00:00 /bin/bash
student  26923 2005  0 19:14 pts/0    00:00:00 ps -f
student  26924 2005  0 19:14 pts/0    00:00:00 tee -a editing_final_lab.txt
```

Summary

Redirecting Output to a File or Program

Describing how program output is displayed, controlled, and saved effectively.

Editing Text Files from the Shell Prompt

Edit files using Vim, a text-based administrator's editing program.

Editing Text Files with a Graphical Editor

Using an editor in a graphical desktop environment to change file content and move text between windows and files.



CHAPTER 5

MANAGING LOCAL LINUX USERS AND GROUPS

Overview	
Goal	To manage local Linux users and groups and administer local password policies.
Objectives	<ul style="list-style-type: none">Explain the role of users and groups on a Linux system and how they are understood by the computer.Run commands as the superuser to administer a Linux system.Create, modify, lock, and delete locally defined user accounts.Create, modify, and delete locally defined group accounts.Lock accounts manually or by setting a password-aging policy in the shadow password file.
Sections	<ul style="list-style-type: none">Users and Groups (and Practice)Gaining Superuser Access (and Practice)Managing Local User Accounts (and Practice)Managing Local Group Accounts (and Practice)Managing User Passwords (and Practice)
Lab	<ul style="list-style-type: none">Managing Local Linux Users and Groups

Users and Groups

Objectives

After completing this section, students should be able to explain the role of users and groups on a Linux system and how they are understood by the computer.

What is a user?

Every process (running program) on the system runs as a particular user. Every file is owned by a particular user. Access to files and directories are restricted by user. The user associated with a running process determines the files and directories accessible to that process.

The **id** command is used to show information about the current logged-in user. Basic information about another user can also be requested by passing in the username of that user as the first argument to the **id** command.

```
[student@desktopX ~]$ id
uid=1000(student) gid=1000(student) groups=1000(student),10(wheel)
context=unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023
```

To view the user associated with a file or directory, use the **ls -l** command. The third column shows the username:

```
[student@serverX ~]$ ls -l /tmp
drwx----- 2 gdm      gdm      4096 Jan 24 13:05 orbit-gdm
drwx----- 2 student  student  4096 Jan 25 20:40 orbit-student
-rw-r--r-- 1 root     root    23574 Jan 24 13:05 postconf
```

To view process information, use the **ps** command. The default is to show only processes in the current shell. Add the **a** option to view all processes with a terminal. To view the user associated with a process, include the **u** option. The first column shows the username:

```
[student@serverX ~]$ ps au
USER      PID %CPU %MEM    VSZ   RSS TTY      STAT START  TIME COMMAND
root      428  0.0  0.7 152768 14400  tty1      Ss+  Feb03  0:04 /usr/bin/Xorg
root      511  0.0  0.0 110012   812  ttyS0      Ss+  Feb03  0:00 /sbin/agetty
root     1805  0.0  0.1 116040  2580  pts/0      Ss  Feb03  0:00 -bash
root     2109  0.0  0.1 178468  2200  pts/0      S  Feb03  0:00 su - student
student   2110  0.0  0.1 116168  2864  pts/0      S  Feb03  0:00 -bash
student   3690  0.0  0.0 123368  1300  pts/0      R+  11:42  0:00 ps au
```

The output of the previous commands displays users by name, but internally the operating system tracks users by a *UID number*. The mapping of names to numbers is defined in databases of account information. By default, systems use a simple "flat file," the **/etc/passwd** file, to store information about local users. The format of **/etc/passwd** follows (seven colon-separated fields):

❶	username:	❷	password:	❸	UID:	❹	GID:	❺	GECOS:	❻	/home/dir:	❻	shell
---	-----------	---	-----------	---	------	---	------	---	--------	---	------------	---	-------

- ❶ *username* is a mapping of a UID to a name for the benefit of human users.

- ❷ *password* is where, historically, passwords were kept in an encrypted format. Today, they are stored in a separate file called **/etc/shadow**.
- ❸ *UID* is a user ID, a number that identifies the user at the most fundamental level.
- ❹ *GID* is the user's primary group ID number. Groups will be discussed in a moment.
- ❺ *GECOS* field is arbitrary text, which usually includes the user's real name.
- ❻ */home/dir* is the location of the user's personal data and configuration files.
- ❼ *shell* is a program that runs as the user logs in. For a regular user, this is normally the program that provides the user's command line prompt.

What is a group?

Like users, groups have a name and a number (GID). Local groups are defined in **/etc/group**.

Primary groups

- Every user has exactly one *primary group*.
- For local users, the primary group is defined by the GID number of the group listed in the third field of **/etc/passwd**.
- Normally, the primary group owns new files created by the user.
- Normally, the primary group of a newly created user is a newly created group with the same name as the user. The user is the only member of this *User Private Group* (UPG).

Supplementary groups

- Users may be a member of zero or more *supplementary groups*.
- The users that are supplementary members of local groups are listed in the last field of the group's entry in **/etc/group**. For local groups, user membership is determined by a comma-separated list of users found in the last field of the group's entry in **/etc/group**:

```
groupname:password:GID:list,of,users,in,this,group
```

- Supplementary group membership is used to help ensure that users have access permissions to files and other resources on the system.

R

References

id(1), **passwd(5)**, and **group(5)** man pages

info libc (*GNU C Library Reference Manual*)

- Section 29: Users and groups

(Note that the *glibc-devel* package must be installed for this info node to be available.)

Practice: User and Group Concepts

Quiz

Match the items below to their counterparts in the table.

/etc/group	/etc/passwd	GID	UID
home directory	login shell	primary group	

Description	Keyword
A number that identifies the user at the most fundamental level	
The program that provides the user's command line prompt	
Location of local group information	
Location of the user's personal files	
A number that identifies the group at the most fundamental level	
Location of local user account information	
The fourth field of /etc/passwd	

Solution

Match the items below to their counterparts in the table.

Description	Keyword
A number that identifies the user at the most fundamental level	UID
The program that provides the user's command line prompt	login shell
Location of local group information	/etc/group
Location of the user's personal files	home directory
A number that identifies the group at the most fundamental level	GID
Location of local user account information	/etc/passwd
The fourth field of /etc/passwd	primary group

Gaining Superuser Access

Objectives

After completing this section, students should be able to run commands as the superuser to administer a Linux system.

The **root** user

Most operating systems have some sort of *superuser*, a user that has all power over the system. This user in Red Hat Enterprise Linux is the **root** user. This user has the power to override normal privileges on the file system, and is used to manage and administer the system. In order to perform tasks such as installing or removing software and to manage system files and directories, a user must escalate privileges to the **root** user.

Most devices can only be controlled by **root**, but there are a few exceptions. For instance, removable devices, such as USB devices, are allowed to be controlled by a normal user. Thus, a non-root user is allowed to add and remove files and otherwise manage a removable device, but only root is allowed to manage "fixed" hard drives by default.

This unlimited privilege, however, comes with responsibility. **root** has unlimited power to damage the system: remove files and directories, remove user accounts, add backdoors, etc. If the **root** account is compromised, someone else would have administrative control of the system. Throughout this course, administrators will be encouraged to log in as a normal user and escalate privileges to **root** only when needed.

The **root** account on Linux is roughly equivalent to the local Administrator account on Windows. In Linux, most system administrators log into an unprivileged user account and use various tools to temporarily gain root privileges.



Warning

One common practice on Windows in the past is for the local Administrator user to log in directly to perform system administrator duties. However, on Linux, it is recommended that system administrators *should not* log in directly as **root**. Instead, system administrators should log in as a non-root user, and use other mechanisms (**su**, **sudo**, or **PolicyKit**, for example) to temporarily gain superuser privileges.

By logging in as the administrative user, the entire desktop environment unnecessarily runs with administrative privileges. In that situation, any security vulnerability which would normally only compromise the user account has the potential to compromise the entire system.

In recent versions of Microsoft Windows, Administrator disabled by default, and features such as User Account Control (UAC) are used to limit administrative privileges for users until actually needed. In Linux, the **PolicyKit** system is the nearest equivalent to UAC.

Switching users with **su**

The **su** command allows a user to switch to a different user account. If a username is not specified, the *root* account is implied. When invoked as a regular user, a prompt will display asking for the password of the account you are switching to; when invoked as *root*, there is no need to enter the account password.

su [-] <username>

```
[student@desktopX ~]$ su -
Password: redhat
[root@desktopX ~]#
```

The command **su username** starts a *non-login shell*, while the command **su - username** starts a *login shell*. The main distinction is **su -** sets up the shell environment as if this were a clean login as that user, while **su** just starts a shell as that user with the current environment settings.

In most cases, administrators want to run **su -** to get the user's normal settings. For more information, see the **bash(1)** man page.



Note

The **su** command is most frequently used to get a command line interface (shell prompt) which is running as another user, typically **root**. However, with the **-c** option, it can be used like the Windows utility **runas** to run an arbitrary program as another user. See **info su** for details.

Running commands as *root* with **sudo**

Fundamentally, Linux implements a very coarse-grained permissions model: *root* can do everything, other users can do nothing (systems-related). The common solution previously discussed is to allow standard users to temporarily “become *root*” using the **su** command. The disadvantage is that while acting as *root*, all the privileges (and responsibilities) of *root* are granted. Not only can the user restart the web server, but they can also remove the entire **/etc** directory. Additionally, all users requiring superuser privilege in this manner must know the **root** password.

The **sudo** command allows a user to be permitted to run a command as *root*, or as another user, based on settings in the **/etc/sudoers** file. Unlike other tools such as **su**, **sudo** requires users to enter their own password for authentication, not the password of the account they are trying to access. This allows an administrator to hand out fine-grained permissions to users to delegate system administration tasks, without having to hand out the *root* password.

For example, when **sudo** has been configured to allow the user *student* to run the command **usermod** as *root*, *student* could run the following command to lock a user account:

```
[student@serverX ~]$ sudo usermod -L username
[sudo] password for student: password
```

One additional benefit to using **sudo** is that all commands executed using **sudo** are logged by default to **/var/log/secure**.

```
[student@serverX ~]$ sudo tail /var/log/secure
...
Feb 19 15:23:36 localhost sudo: student : TTY=pts/0 ; PWD=/home/student ; USER=root ;
COMMAND=/sbin/usermod -L student
Feb 19 15:23:36 localhost usermod[16325]: lock user 'student' password
Feb 19 15:23:47 localhost sudo: student : TTY=pts/0 ; PWD=/home/student ; USER=root ;
COMMAND=/bin/tail /var/log/secure
```

In Red Hat Enterprise Linux 7, all members of group **wheel** can use **sudo** to run commands as any user, including **root**. The user will be prompted for their own password. This is a change from Red Hat Enterprise Linux 6 and earlier. Users who were members of group **wheel** did not get this administrative access by default in RHEL 6 and earlier.

To enable similar behavior on earlier versions of Red Hat Enterprise Linux, use **visudo** to edit the configuration file and uncomment the line allowing the group **wheel** to run all commands.

```
[root@desktopX ~]# cat /etc/sudoers
...Output omitted...
## Allows people in group wheel to run all commands
%wheel    ALL=(ALL)        ALL

## Same thing without a password
# %wheel  ALL=(ALL)      NOPASSWD: ALL
...Output omitted...
```



Warning

RHEL 6 did not grant group **wheel** any special privileges by default. Sites which have been using this group may be surprised when RHEL 7 automatically grants all members of **wheel** full **sudo** privileges. This could lead to unauthorized users getting superuser access to RHEL 7 systems.

Historically, membership in group **wheel** has been used by Unix-like systems to grant or control superuser access.

Most system administration applications with a GUI use **PolicyKit** to prompt users for authentication and to manage root access. In Red Hat Enterprise Linux 7, **PolicyKit** may also prompt members of group **wheel** for their own password in order to get **root** privileges when using graphical tools. This is similar to the way in which they can use **sudo** to get those privileges at the shell prompt. **PolicyKit** grants these privileges based on its own configuration settings, separate from **sudo**. Advanced students may be interested in the **pkexec(1)** and **polkit(8)** man pages for details on how this system works, but it is beyond the scope of this course.



References

su(1) and **sudo(8)** man pages

info libc (*GNU C Library Reference Manual*)

- Section 29.2: The Persona of a Process

(Note that the *glibc-devel* package must be installed for this info node to be available.)

Practice: Running Commands as root

Guided exercise

In this lab, you will practice running commands as **root**.

Outcomes

Use the **su** with and without login scripts to switch users. Use **sudo** to run commands with privilege.

Before you begin...

Reset your serverX system.

- 1. Log into the GNOME desktop on serverX as **student** with a password of **student**.

- 2. Open a window with a Bash prompt.

Select Applications > Utilities > Terminal.

- 3. Explore characteristics of the current student login environment.

- 3.1. View the user and group information and display the current working directory.

```
[student@serverX ~]$ id  
uid=1000(student) gid=1000(student) groups=1000(student),10(wheel)  
context=unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023  
[student@serverX ~]$ pwd  
/home/student
```

- 3.2. View the variables which specify the home directory and the locations searched for executable files.

```
[student@serverX ~]$ echo $HOME  
/home/student  
[student@serverX ~]$ echo $PATH  
/usr/local/bin:/usr/local/sbin:/usr/bin:/usr/sbin:/bin:/sbin:/home/  
student/.local/bin:/home/student/bin
```

- 4. Switch to root without the dash and explore characteristics of the new environment.

- 4.1. Become the **root** user at the shell prompt.

```
[student@serverX ~]$ su  
Password: redhat
```

- 4.2. View the user and group information and display the current working directory.
Note the identity changed, but not the current working directory.

```
[root@serverX student]# id  
uid=0(root) gid=0(root) groups=0(root)  
context=unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023  
[root@serverX student]# pwd
```

```
/home/student
```

- 4.3. View the variables which specify the home directory and the locations searched for executable files. Look for references to the student and root accounts.

```
[root@serverX student]# echo $HOME  
/root  
[root@serverX student]# echo $PATH  
/usr/local/bin:/usr/local/sbin:/usr/bin:/usr/sbin:/bin:/sbin:/home/  
student/.local/bin:/home/student/bin
```

- 4.4. Exit the shell to return to the **student** user.

```
[root@serverX student]# exit  
exit
```

- 5. Switch to root with the dash and explore characteristics of the new environment.
 - 5.1. Become the **root** user at the shell prompt. Be sure all the login scripts are also executed.

```
[student@serverX ~]$ su -  
Password: redhat
```

- 5.2. View the user and group information and display the current working directory.

```
[root@serverX ~]# id  
uid=0(root) gid=0(root) groups=0(root)  
context=unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023  
[root@serverX ~]# pwd  
/root
```

- 5.3. View the variables which specify the home directory and the locations searched for executable files. Look for references to the student and root accounts.

```
[root@serverX ~]# echo $HOME  
/root  
[root@serverX ~]# echo $PATH  
/usr/local/sbin:/usr/local/bin:/sbin:/bin:/usr/sbin:/usr/bin:/root/bin
```

- 5.4. Exit the shell to return to the **student** user.

```
[root@serverX ~]# exit  
logout
```

- 6. Run several commands as student which require root access.
 - 6.1. View the last 5 lines of the **/var/log/messages**.

```
[student@serverX ~]$ tail -5 /var/log/messages  
tail: cannot open '/var/log/messages' for reading: Permission denied
```

```
[student@serverX ~]$ sudo tail -5 /var/log/messages
Feb  3 15:07:22 localhost su: (to root) root on pts/0
Feb  3 15:10:01 localhost systemd: Starting Session 31 of user root.
Feb  3 15:10:01 localhost systemd: Started Session 31 of user root.
Feb  3 15:12:05 localhost su: (to root) root on pts/0
Feb  3 15:14:47 localhost su: (to student) root on pts/0
```

- 6.2. Make a backup of a configuration file in the **/etc** directory.

```
[student@serverX ~]$ cp /etc/motd /etc/motdOLD
cp: cannot create regular file '/etc/motdOLD': Permission denied
[student@serverX ~]$ sudo cp /etc/motd /etc/motdOLD
```

- 6.3. Remove the **/etc/motdOLD** file that was just created.

```
[student@serverX ~]$ rm /etc/motdOLD
rm: remove write-protected regular empty file '/etc/motdOLD'? y
rm: cannot remove '/etc/motdOLD': Permission denied
[student@serverX ~]$ sudo rm /etc/motdOLD
```

- 6.4. Edit a configuration file in the **/etc** directory.

```
[student@serverX ~]$ echo "Welcome to class" >> /etc/motd
-bash: /etc/motd: Permission denied
[student@serverX ~]$ sudo vim /etc/motd
```

Managing Local User Accounts

Objectives

After completing this section, students should be able to create, modify, lock, and delete locally defined user accounts.

Managing local users

A number of command-line tools can be used to manage local user accounts.

useradd creates users

- **useradd *username*** sets reasonable defaults for all fields in **/etc/passwd** when run without options. The **useradd** command does not set any valid password by default, and the user cannot log in until a password is set.
- **useradd --help** will display the basic options that can be used to override the defaults. In most cases, the same options can be used with the **usermod** command to modify an existing user.
- Some defaults, such as the range of valid UID numbers and default password aging rules, are read from the **/etc/login.defs** file. Values in this file are only used when creating new users. A change to this file will not have an effect on any existing users.

usermod modifies existing users

- **usermod --help** will display the basic options that can be used to modify an account. Some common options include:

usermod options:	
-c, --comment COMMENT	Add a value, such as a full name, to the GECOS field.
-g, --gid GROUP	Specify the primary group for the user account.
-G, --groups GROUPS	Specify a list of supplementary groups for the user account.
-a, --append	Used with the -G option to append the user to the supplemental groups mentioned without removing the user from other groups.
-d, --home HOME_DIR	Specify a new home directory for the user account.
-m, --move-home	Move a user home directory to a new location. Must be used with the -d option.
-s, --shell SHELL	Specify a new login shell for the user account.
-L, --lock	Lock a user account.
-U, --unlock	Unlock a user account.

userdel deletes users

- **userdel *username*** removes the user from **/etc/passwd**, but leaves the home directory intact by default.
- **userdel -r *username*** removes the user and the user's home directory.



Warning

When a user is removed with **userdel** without the **-r** option specified, the system will have files that are owned by an unassigned user ID number. This can also happen when files created by a deleted user exist outside their home directory. This situation can lead to information leakage and other security issues.

In Red Hat Enterprise Linux 7 the **useradd** command assigns new users the first free UID number available in the range starting from UID 1000 or above. (unless one is explicitly specified with the **-u *UID*** option). This is how information leakage can occur: If the first free UID number had been previously assigned to a user account which has since been removed from the system, the old user's UID number will get reassigned to the new user, giving the new user ownership of the old user's remaining files. The following scenario demonstrates this situation:

```
[root@serverX ~]# useradd prince
[root@serverX ~]# ls -l /home
drwx----- 3 prince prince 74 Feb 4 15:22 prince
[root@serverX ~]# userdel prince
[root@serverX ~]# ls -l /home
drwx----- 3 1000 1000 74 Feb 4 15:22 prince
[root@serverX ~]# useradd bob
[root@serverX ~]# ls -l /home
drwx----- 3 bob bob 74 Feb 4 15:23 bob
drwx----- 3 bob bob 74 Feb 4 15:22 prince
```

Notice that **bob** now owns all files that **prince** once owned. Depending on the situation, one solution to this problem is to remove all "unowned" files from the system when the user that created them is deleted. Another solution is to manually assign the "unowned" files to a different user. The root user can find "unowned" files and directories by running: **find / -nouser -o -nogroup 2> /dev/null**.

id displays user information

- **id** will display user information, including the user's UID number and group memberships.
- **id *username*** will display user information for *username*, including the user's UID number and group memberships.

passwd sets passwords

- **passwd *username*** can be used to either set the user's initial password or change that user's password.
- The **root** user can set a password to any value. A message will be displayed if the password does not meet the minimum recommended criteria, but is followed by a prompt to retype the new password and all tokens are updated successfully.

```
[root@serverX ~]# passwd student
Changing password for user student.
New password: redhat123
BAD PASSWORD: The password fails the dictionary check - it is based on a dictionary word
```

```
Retype new password: redhat123
passwd: all authentication tokens updated successfully.
```

- A regular user must choose a password which is at least 8 characters in length and is not based on a dictionary word, the username, or the previous password.

UID ranges

Specific UID numbers and ranges of numbers are used for specific purposes by Red Hat Enterprise Linux.

- *UID 0* is always assigned to the superuser account, **root**.
- *UID 1-200* is a range of "system users" assigned statically to system processes by Red Hat.
- *UID 201-999* is a range of "system users" used by system processes that do not own files on the file system. They are typically assigned dynamically from the available pool when the software that needs them is installed. Programs run as these "unprivileged" system users in order to limit their access to just the resources they need to function.
- *UID 1000+* is the range available for assignment to regular users.



Note

Prior to Red Hat Enterprise Linux 7, the convention was that UID 1-499 was used for system users and UID 500+ for regular users. Default ranges used by **useradd** and **groupadd** can be changed in the **/etc/login.defs** file.

R

References

useradd(8), **usermod(8)**, **userdel(8)** man pages

Practice: Creating Users Using Command-line Tools

Guided exercise

In this lab, you will create a number of users on your serverX system, setting and recording an initial password for each user.

Outcomes

A system with additional user accounts.

Before you begin...

Reset your serverX system.

- 1. Log into the GNOME desktop on serverX as **student** with a password of **student**.
- 2. Open a window with a Bash prompt.
Select Applications > Utilities > Terminal.
- 3. Become the **root** user at the shell prompt.

```
[student@serverX ~]$ su -  
Password: redhat
```

- 4. Add the user *juliet*.

```
[root@serverX ~]# useradd juliet
```

- 5. Confirm that *juliet* has been added by examining the **/etc/passwd** file.

```
[root@serverX ~]# tail -2 /etc/passwd  
tcpdump:x:72:72:::/sbin/nologin  
juliet:x:1001:1001::/home/juliet:/bin/bash
```

- 6. Use the **passwd** command to initialize *juliet*'s password.

```
[root@serverX ~]# passwd juliet  
Changing password for user juliet.  
New password: juliet  
BAD PASSWORD: The password is shorter than 8 characters  
Retype new password: juliet  
passwd: all authentication tokens updated successfully.
```

- 7. Continue adding the remaining users in the steps below and set initial passwords.

- 7.1. romeo

```
[root@serverX ~]# useradd romeo
```

```
[root@serverX ~]# passwd romeo
Changing password for user romeo.
New password: romeo
BAD PASSWORD: The password is shorter than 8 characters
Retype new password: romeo
passwd: all authentication tokens updated successfully.
```

□ 7.2. hamlet

```
[root@serverX ~]# useradd hamlet
[root@serverX ~]# passwd hamlet
```

□ 7.3. reba

```
[root@serverX ~]# useradd reba
[root@serverX ~]# passwd reba
```

□ 7.4. dolly

```
[root@serverX ~]# useradd dolly
[root@serverX ~]# passwd dolly
```

□ 7.5. elvis

```
[root@serverX ~]# useradd elvis
[root@serverX ~]# passwd elvis
```

Managing Local Group Accounts

Objectives

After completing this section, students should be able to create, modify, and delete locally defined group accounts.

Managing supplementary groups

A group must exist before a user can be added to that group. Several command-line tools are used to manage local group accounts.

groupadd creates groups

- **groupadd *groupname*** without options uses the next available GID from the range specified in the **/etc/login.defs** file.
- The **-g *GID*** option is used to specify a specific GID.

```
[student@serverX ~]$ sudo groupadd -g 5000 ateam
```



Note

Given the automatic creation of user private groups (GID 1000+), it is generally recommended to set aside a range of GID numbers to be used for supplementary groups. A higher range will avoid a collision with a system group (GID 0-999).

- The **-r** option will create a system group using a GID from the range of valid system GID numbers listed in the **/etc/login.defs** file.

```
[student@serverX ~]$ sudo groupadd -r appusers
```

groupmod modifies existing groups

- The **groupmod** command is used to change a group name to a GID mapping. The **-n** option is used to specify a new name.

```
[student@serverX ~]$ sudo groupmod -n javaapp appusers
```

- The **-g** option is used to specify a new GID.

```
[student@serverX ~]$ sudo groupmod -g 6000 ateam
```

groupdel deletes a group

- The **groupdel** command will remove a group.

```
[student@serverX ~]$ sudo groupdel javaapp
```

- A group may not be removed if it is the primary group of any existing user. As with **userdel**, check all file systems to ensure that no files remain owned by the group.

usermod alters group membership

- The membership of a group is controlled with user management. Change a user's primary group with **usermod -g groupname**.

```
[student@serverX ~]$ sudo usermod -g student student
```

- Add a user to a supplementary group with **usermod -aG groupname username**.

```
[student@serverX ~]$ sudo usermod -aG wheel elvis
```



Important

The use of the **-a** option makes **usermod** function in "append" mode. Without it, the user would be removed from *all other* supplementary groups.

R

References

group(5), **groupadd(8)**, **groupdel(8)**, and **usermod(8)** man pages

Practice: Managing Groups Using Command-line Tools

Guided exercise

In this lab, you will add users to newly created supplementary groups.

Outcomes

The **shakespeare** group consists of **juliet**, **romeo**, and **hamlet**. The **artists** group contains **reba**, **dolly**, and **elvis**.

Before you begin...

Perform the following steps on serverX unless directed otherwise.

- 1. Become the **root** user at the shell prompt.

```
[student@serverX ~]$ su -  
Password: redhat
```

- 2. Create a supplementary group called **shakespeare** with a group ID of **30000**.

```
[root@serverX ~]# groupadd -g 30000 shakespeare
```

- 3. Create a supplementary group called **artists**.

```
[root@serverX ~]# groupadd artists
```

- 4. Confirm that *shakespeare* and *artists* have been added by examining the **/etc/group** file.

```
[root@serverX ~]# tail -5 /etc/group  
reba:x:1004:  
dolly:x:1005:  
elvis:x:1006:  
shakespeare:x:30000:  
artists:x:30001:
```

- 5. Add the *juliet* user to the *shakespeare* group as a supplementary group.

```
[root@serverX ~]# usermod -G shakespeare juliet
```

- 6. Confirm that *juliet* has been added using the **id** command.

```
[root@serverX ~]# id juliet  
uid=1001(juliet) gid=1001(juliet) groups=1001(juliet),30000(shakespeare)
```

- 7. Continue adding the remaining users to groups as follows:

- 7.1. Add *romeo* and *hamlet* to the *shakespeare* group.

```
[root@serverX ~]# usermod -G shakespeare romeo
[root@serverX ~]# usermod -G shakespeare hamlet
```

- 7.2. Add *reba*, *dolly*, and *elvis* to the *artists* group.

```
[root@serverX ~]# usermod -G artists reba
[root@serverX ~]# usermod -G artists dolly
[root@serverX ~]# usermod -G artists elvis
```

- 7.3. Verify the supplemental group memberships by examining the **/etc/group** file.

```
[root@serverX ~]# tail -5 /etc/group
reba:x:1004:
dolly:x:1005:
elvis:x:1006:
shakespeare:x:30000:juliet,romeo,hamlet
artists:x:30001:reba,dolly,elvis
```

Managing User Passwords

Objectives

After completing this section, students should be able to lock accounts manually or by setting a password-aging policy in the shadow password file.

Shadow passwords and password policy

In the distant past, encrypted passwords were stored in the world-readable `/etc/passwd` file. This was thought to be reasonably secure until dictionary attacks on encrypted passwords became common. At that point, the encrypted passwords, or "password hashes," were moved to the more secure `/etc/shadow` file. This new file also allowed password aging and expiration features to be implemented.

There are three pieces of information stored in a modern password hash:

\$1\$gCjLa2/Z\$6Pu0EK0AzfCjxjv2hoLOB/

1. **1:** The hashing algorithm. The number 1 indicates an MD5 hash. The number 6 appears when a SHA-512 hash is used.
2. **gCjLa2/Z:** The *salt* used to encrypt the hash. This is originally chosen at random. The salt and the unencrypted password are combined and encrypted to create the encrypted password hash. The use of a salt prevents two users with the same password from having identical entries in the `/etc/shadow` file.
3. **6Pu0EK0AzfCjxjv2hoLOB/:** The encrypted hash.

When a user tries to log in, the system looks up the entry for the user in `/etc/shadow`, combines the salt for the user with the unencrypted password that was typed in, and encrypts them using the hashing algorithm specified. If the result matches the encrypted hash, the user typed in the right password. If the result doesn't match the encrypted hash, the user typed in the wrong password and the login attempt fails. This method allows the system to determine if the user typed in the correct password without storing that password in a form usable for logging in.



Note

Red Hat Enterprise Linux 6 and 7 support two new strong password hashing algorithms, SHA-256 (algorithm **5**) and SHA-512 (algorithm **6**). Both the salt string and the encrypted hash are longer for these algorithms. The default algorithm used for password hashes can be changed by the **root** user by running the command **authconfig --passalgo** with one of the arguments **md5**, **sha256**, or **sha512**, as appropriate.

Red Hat Enterprise Linux 7 defaults to using SHA-512 encryption.

/etc/shadow format

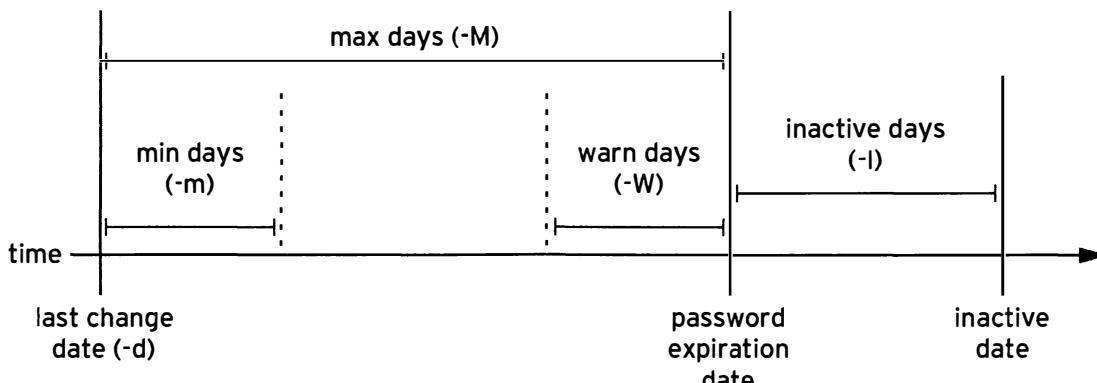
The format of `/etc/shadow` follows (nine colon-separated fields):

1 name:	2 password:	3 lastchange:	4 minage:	5 maxage:	6 warning:	7 inactive:	8 expire:	9 blank
----------------	--------------------	----------------------	------------------	------------------	-------------------	--------------------	------------------	----------------

- ➊ The login *name*. This must be a valid account name on the system.
- ➋ The encrypted *password*. A password field which starts with a exclamation mark means that the password is locked.
- ➌ The date of the last *password change*, represented as the number of days since 1970.01.01.
- ➍ The *minimum* number of days before a password may be changed, where 0 means "no minimum age requirement."
- ➎ The *maximum* number of days before a password must be changed.
- ➏ The *warning* period that a password is about to expire. Represented in days, where 0 means "no warning given."
- ➐ The number of days an account remains active after a password has expired. A user may still log into the system and change the password during this period. After the specified number of days, the account is locked, becoming *inactive*.
- ➑ The account *expiration* date, represented as the number of days since 1970.01.01.
- ➒ This *blank* field is reserved for future use.

Password aging

The following diagram relates the relevant password-aging parameters, which can be adjusted using **chage** to implement a password-aging policy.



chage -d 0 username will force a password update on next login.

chage -l username will list a username's current settings.

chage -E YYYY-MM-DD will expire an account on a specific day.



Note

The **date** command can be used to calculate a date in the future.

```
[student@serverX ~]$ date -d "+45 days"
Sat Mar 22 11:47:06 EDT 2014
```

Restricting access

With the **chage** command, an account expiration can be set. Once that date is reached, the user cannot log into the system interactively. The **usermod** command can "lock" an account with the **-L** option.

```
[student@serverX ~]$ sudo usermod -L elvis
[student@serverX ~]$ su - elvis
Password: elvis
su: Authentication failure
```

When a user has left the company, the administrator may lock and expire an account with a single **usermod** command. The date must be given as the number of days since 1970.01.01.

```
[student@serverX ~]$ sudo usermod -L -e 1 elvis
```

Locking the account prevents the user from authenticating with a password to the system. It is the recommended method of preventing access to an account by an employee who has left the company. If the employee returns, the account can later be unlocked with **usermod -U USERNAME**. If the account was also expired, be sure to also change the expiration date.

The **nologin** shell

Sometimes a user needs an account with a password to authenticate to a system, but does not need an interactive shell on the system. For example, a mail server may require an account to store mail and a password for the user to authenticate with a mail client used to retrieve mail. That user does not need to log directly into the system.

A common solution to this situation is to set the user's login shell to **/sbin/nologin**. If the user attempts to log into the system directly, the **nologin** "shell" will simply close the connection.

```
[root@serverX ~]# usermod -s /sbin/nologin student
[root@serverX ~]# su - student
Last login: Tue Feb  4 18:40:30 EST 2014 on pts/0
This account is currently not available.
```



Important

Use of the **nologin** shell prevents interactive use of the system, but does not prevent all access. A user may still be able to authenticate and upload or retrieve files through applications such as web applications, file transfer programs, or mail readers.

R

References

chage(1), **usermod(8)**, **shadow(5)**, **crypt(3)** man pages

Practice: Managing User Password Aging

Guided exercise

In this lab, you will set unique password policies for users.

Outcomes

The password for **romeo** must be changed when the user first logs into the system, every 90 days thereafter, and the account expires in 180 days.

Before you begin...

Perform the following steps on serverX unless directed otherwise.

- 1. Explore locking and unlocking accounts.

- 1.1. Lock the **romeo** account.

```
[student@serverX ~]$ sudo usermod -L romeo
```

- 1.2. Attempt to log in as **romeo**.

```
[student@serverX ~]$ su - romeo  
Password: romeo  
su: Authentication failure
```

- 1.3. Unlock the **romeo** account.

```
[student@serverX ~]$ sudo usermod -U romeo
```

- 2. Change the password policy for **romeo** to require a new password every 90 days.

```
[student@serverX ~]$ sudo chage -M 90 romeo  
[student@serverX ~]$ sudo chage -l romeo  
Last password change : Feb 03, 2014  
Password expires : May 04, 2014  
Password inactive : never  
Account expires : never  
Minimum number of days between password change : 0  
Maximum number of days between password change : 90  
Number of days of warning before password expires : 7
```

- 3. Additionally, force a password change on the first login for the **romeo** account.

```
[student@serverX ~]$ sudo chage -d 0 romeo
```

- 4. Log in as **romeo** and change the password to **forsooth123**.

```
[student@serverX ~]$ su - romeo  
Password: romeo  
You are required to change your password immediately (root enforced)
```

```
Changing password for romeo.  
(current) UNIX password: romeo  
New password: forsooth123  
Retype new password: forsooth123  
[romeo@serverX ~]$ exit
```

- 5. Expire accounts in the future.
 - 5.1. Determine a date 180 days in the future.

```
[student@serverX ~]$ date -d "+180 days"  
Sat Aug 2 17:05:20 EDT 2014
```

- 5.2. Set accounts to expire on that date.

```
[student@serverX ~]$ sudo chage -E 2014-08-02 romeo  
[student@serverX ~]$ sudo chage -l romeo  
Last password change : Feb 03, 2014  
Password expires     : May 04, 2014  
Password inactive   : never  
Account expires      : Aug 02, 2014  
Minimum number of days between password change : 0  
Maximum number of days between password change : 90  
Number of days of warning before password expires : 7
```

Lab: Managing Local Linux Users and Groups

Performance checklist

In this lab, you will define a default password policy, create a supplementary group of three new users, and modify the password policy of one user.

Outcomes

- A new group on serverX called **consultants**, including three new user accounts for Sam Spade, Betty Boop, and Dick Tracy.
- All new accounts should require that passwords be changed at first login and every 30 days thereafter.
- The new consultant accounts should expire at the end of the 90-day contract, and Betty Boop must change her password every 15 days.

Before you begin...

Reset your serverX system.

1. Ensure that newly created users have passwords which must be changed every 30 days.
2. Create a new group named **consultants** with a GID of 40000.
3. Create three new users: **ssspade**, **bboop**, and **dtracy**, with a password of **default** and add them to the supplementary group **consultants**. The primary group should remain as the user private group.
4. Determine the date 90 days in the future and set each of the three new user accounts to expire on that date.
5. Change the password policy for the **bboop** account to require a new password every 15 days.
6. Additionally, force all users to change their password on first login.
7. When you finish, run the **lab localusers grade** evaluation script to confirm you have done everything correctly.

Solution

In this lab, you will define a default password policy, create a supplementary group of three new users, and modify the password policy of one user.

Outcomes

- A new group on serverX called **consultants**, including three new user accounts for Sam Spade, Betty Boop, and Dick Tracy.
- All new accounts should require that passwords be changed at first login and every 30 days thereafter.
- The new consultant accounts should expire at the end of the 90-day contract, and Betty Boop must change her password every 15 days.

Before you begin...

Reset your serverX system.

1. Ensure that newly created users have passwords which must be changed every 30 days.

```
[student@serverX ~]$ sudo vim /etc/login.defs
[student@serverX ~]$ cat /etc/login.defs
...Output omitted...
PASS_MAX_DAYS 30
PASS_MIN_DAYS 0
PASS_MIN_LEN 5
PASS_WARN_AGE 7
...Output omitted...
```

2. Create a new group named **consultants** with a GID of 40000.

```
[student@serverX ~]$ sudo groupadd -g 40000 consultants
[student@serverX ~]$ tail -5 /etc/group
stapdev:x:158:
pesign:x:989:
tcpdump:x:72:
slocate:x:21:
consultants:x:40000:
```

3. Create three new users: **ssspade**, **bboop**, and **dtracy**, with a password of **default** and add them to the supplementary group **consultants**. The primary group should remain as the user private group.

```
[student@serverX ~]$ sudo useradd -G consultants sspade
[student@serverX ~]$ sudo useradd -G consultants bboop
[student@serverX ~]$ sudo useradd -G consultants dtracy
[student@serverX ~]$ tail -5 /etc/group
slocate:x:21:
consultants:x:40000:ssspade,bboop,dtracy
ssspade:x:1001:
bboop:x:1002:
dtracy:x:1003:
[student@serverX ~]$ sudo passwd sspade
Changing password for user sspade.
New password: default
BAD PASSWORD: The password is shorter than 8 characters
```

```
Retype new password: default
passwd: all authentication tokens updated successfully.
[student@serverX ~]$ sudo passwd bboop
[student@serverX ~]$ sudo passwd dtracy
```

4. Determine the date 90 days in the future and set each of the three new user accounts to expire on that date.

```
[student@serverX ~]$ date -d "+90 days"
Mon May  5 11:49:24 EDT 2014
[student@serverX ~]$ sudo chage -E 2014-05-05 sspade
[student@serverX ~]$ sudo chage -E 2014-05-05 bboop
[student@serverX ~]$ sudo chage -E 2014-05-05 dtracy
```

5. Change the password policy for the **bboop** account to require a new password every 15 days.

```
[student@serverX ~]$ sudo chage -M 15 bboop
[student@serverX ~]$ sudo chage -l bboop
Last password change : Feb 04, 2014
Password expires     : Feb 19, 2014
Password inactive   : never
Account expires      : May 05, 2014
Minimum number of days between password change : 0
Maximum number of days between password change : 15
Number of days of warning before password expires : 7
```

6. Additionally, force all users to change their password on first login.

```
[student@serverX ~]$ sudo chage -d 0 sspade
[student@serverX ~]$ sudo chage -d 0 bboop
[student@serverX ~]$ sudo chage -d 0 dtracy
```

7. When you finish, run the **lab localusers grade** evaluation script to confirm you have done everything correctly.

```
[student@serverX ~]$ lab localusers grade
```

Summary

Users and Groups

List the roles of users and groups on a Linux system and view the local configuration files.

Gaining Superuser Access

Escalate privilege to run commands as the superuser.

Managing Local User Accounts

Add, remove, and modify local users with command-line tools.

Managing Local Group Accounts

Manage local groups with command-line tools.

Managing User Passwords

Manage password aging policies for users and manually lock, unlock, and expire accounts.



CHAPTER 6

CONTROLLING ACCESS TO FILES WITH LINUX FILE SYSTEM PERMISSIONS

Overview	
Goal	To set Linux file system permissions on files and interpret the security effects of different permission settings.
Objectives	<ul style="list-style-type: none">Explain how the Linux file permissions model works.Change the permissions and ownership of files using command-line tools.Configure a directory in which newly created files are automatically writable by members of the group which owns the directory, using special permissions and default umask settings.
Sections	<ul style="list-style-type: none">Linux File System Permissions (and Practice)Managing File System Permissions from the Command Line (and Practice)Managing Default Permissions and File Access (and Practice)
Lab	<ul style="list-style-type: none">Controlling Access to Files with Linux File System Permissions

Linux File System Permissions

Objectives

After completing this section, students should be able to explain how the Linux file permissions model works.

Linux file system permissions

Access to files by users are controlled by *file permissions*. The Linux file permissions system is simple but flexible, which makes it easy to understand and apply, yet able to handle most normal permission cases easily.

Files have just three categories of user to which permissions apply. The file is owned by a *user*, normally the one who created the file. The file is also owned by a single *group*, usually the primary group of the user who created the file, but this can be changed. Different permissions can be set for the owning user, the owning group, and for all *other* users on the system that are not the user or a member of the owning group.

The most specific permissions apply. So, *user* permissions override *group* permissions, which override *other* permissions.

In the graphic that follows, joshua is a member of the groups joshua and web, while allison is a member of allison, wheel, and web. When joshua and allison have the need to collaborate, the files should be associated with the group web and the group permissions should allow the desired access.

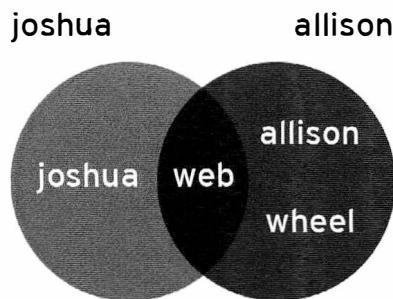


Figure 6.1: Group membership illustration

There are also just three categories of permissions which apply: *read*, *write*, and *execute*. These permissions affect access to files and directories as follows:

Effects of permissions on files and directories

Permission	Effect on files	Effect on directories
r (read)	Contents of the file can be read.	Contents of the directory (file names) can be listed.
w (write)	Contents of the file can be changed.	Any file in the directory may be created or deleted.
x (exec)	Files can be executed as commands.	Contents of the directory can be accessed (dependent on the permissions of the files in the directory).

Note that users normally have both **read** and **exec** on read-only directories, so that they can list the directory and access its contents. If a user only has **read** access on a directory, the names of the files in it can be listed, but no other information, including permissions or time stamps, are available, nor can they be accessed. If a user only has **exec** access on a directory, they cannot list the names of the files in the directory, but if they already know the name of a file which they have permission to read, then they can access the contents of that file by explicitly specifying the file name.

A file may be removed by anyone who has write permission to the directory in which the file resides, regardless of the ownership or permissions on the file itself. (This can be overridden with a special permission, the *sticky bit*, which will be discussed at the end of the unit.)

Viewing file/directory permissions and ownership

The **-l** option of the **ls** command will expand the file listing to include both the permissions of a file and the ownership:

```
[student@desktopX ~]$ ls -l test  
-rw-rw-r--. 1 student student 0 Feb 8 17:36 test
```

The command **ls -l directoryname** will show the expanded listing of all of the files that reside inside the directory. To prevent the descent into the directory and see the expanded listing of the directory itself, add the **-d** option to ls:

```
[student@desktopX ~]$ ls -ld /home  
drwxr-xr-x. 5 root root 4096 Jan 31 22:00 /home
```



Note

Unlike NTFS permissions, Linux permissions only apply to the directory or file that they are set on. Permissions on a directory are not inherited automatically by the subdirectories and files within it. (The permissions on a directory *may* effectively block access to its contents, however.) All permissions in Linux are set directly on each file or directory.

The **read** permission on a directory in Linux is roughly equivalent to **List folder contents** in Windows.

The **write** permission on a directory in Linux is equivalent to **Modify** in Windows; it implies the ability to delete files and subdirectories. In Linux, if **write** and the **sticky bit** are both set on a directory, then only the user that owns a file or subdirectory in the directory may delete it, which is close to the behavior of the Windows **Write** permission.

Root has the equivalent of the Windows **Full Control** permission on all files in Linux. However, root may still have access restricted by the system's SELinux policy and the security context of the process and files in question. SELinux will be discussed in a later course.

Examples: Linux user, group, other concepts

Users and their groups:

<pre> lucy lucy,ricardo ricky ricky,ricardo ethel ethel,mertz fred fred,mertz </pre> <p>File attributes (permissions, user & group ownership, name):</p> <pre> drwxrwxr-x ricky ricardo dir (which contains the following files) -rw-rw-r-- lucy lucy lfile1 -rw-r--rw- lucy ricardo lfile2 -rw-rw-r-- ricky ricardo rfile1 -rw-r----- ricky ricardo rfile2 </pre>
--

Allowed/denied behavior	Controlling permissions
lucy is the only person who can change the contents of lfile1 .	lucy has write permissions on the file lfile1 as the owner. No one is listed as a member of the group lucy . The permissions for <i>other</i> do not include write permissions.
ricky can view the contents of lfile2 , but cannot modify the contents of lfile2 .	ricky is a member of the group ricardo , and that group has read-only permissions on lfile2 . Even though <i>other</i> has write permissions, group permissions take precedence.
ricky can delete lfile1 and lfile2 .	ricky has write permissions on the directory containing both files, and as such, he can delete any file in that directory.
ethel can change the contents of lfile2 .	Since ethel is not lucy , and is not a member of the ricardo group, <i>other</i> permissions apply to her, and those include write permission.
lucy can change the contents of rfile1 .	lucy is a member of the ricardo group, and that group has both read and write permissions on rfile1 .
ricky can view and modify the contents of rfile2 .	ricky owns the file and has both read and write access to rfile2 .
lucy can view but not modify the contents of rfile2 .	lucy is a member of the ricardo group, and that group has read-only access to rfile2 .
ethel and fred do not have any access to the contents of rfile2 .	<i>other</i> permissions apply to ethel and fred , and those permissions do not include read or write permission.



References

ls(1) man page

info coreutils (GNU Coreutils)

- Section 13: Changing file attributes

Practice: Interpreting File and Directory Permissions

Quiz

Using the directory listing presented, match the items that follow to their counterparts in the table.

Users and their groups:

```
wilma    wilma, flintstone
fred     fred, flintstone
betty    betty, rubble
barney   barney, rubble
```

File attributes (permissions, user & group ownership, name):

```
drwxrwxr-x  fred    flintstone  dir (which contains the following files)
-rw-rw-r--  wilma  wilma      lfile1
-rw-r--rw-  wilma  flintstone  lfile2
-rw-rw-r--  fred    flintstone  rfile1
-rw-r----- fred    flintstone  rfile2
```

all	lfile1	lfile2	none	rfile1	rfile2
-----	--------	--------	------	--------	--------

Description	File name
Is owned by fred and readable by all users	
Contents may be modified by the user betty	
Can be deleted by the user fred	
Cannot be read by the user barney	
Has a group ownership of wilma	
Can be deleted by the user barney	

Solution

Using the directory listing presented, match the items that follow to their counterparts in the table.

Users and their groups:

```
wilma    wilma, flintstone
fred     fred, flintstone
betty   betty, rubble
barney  barney, rubble
```

File attributes (permissions, user & group ownership, name):

```
drwxrwxr-x  fred    flintstone  dir (which contains the following files)
-rw-rw-r--  wilma  wilma      lfile1
-rw-r--rw-  wilma  flintstone lfile2
-rw-rw-r--  fred    flintstone rfile1
-rw-r----- fred    flintstone rfile2
```

Description	File name
Is owned by fred and readable by all users	rfile1
Contents may be modified by the user betty	lfile2
Can be deleted by the user fred	all
Cannot be read by the user barney	rfile2
Has a group ownership of wilma	lfile1
Can be deleted by the user barney	none

Managing File System Permissions from the Command Line

Objectives

After completing this section, students should be able to change the permissions and ownership of files using command-line tools.

Changing file/directory permissions

The command used to change permissions from the command line is **chmod**, short for "change mode" (permissions are also called the *mode* of a file). The **chmod** command takes a permission instruction followed by a list of files or directories to change. The permission instruction can be issued either symbolically (the symbolic method) or numerically (the numeric method).

Symbolic method keywords:

```
chmod WhoWhatWhich file|directory
```

- *Who* is u, g, o, a (*for user, group, other, all*)
- *What* is +, -, = (*for add, remove, set exactly*)
- *Which* is r, w, x (*for read, write, executable*)

The *symbolic* method of changing file permissions uses letters to represent the different groups of permissions: **u** for user, **g** for group, **o** for other, and **a** for all.

With the symbolic method, it is not necessary to set a complete new group of permissions. Instead, it is possible to change one or more of the existing permissions. In order to accomplish this, use three symbols: + to add permissions to a set, - to remove permissions from a set, and = to replace the entire set for a group of permissions.

The permissions themselves are represented by a single letter: **r** for read, **w** for write, and **x** for execute.

Numeric method:

```
chmod ### file|directory
```

- Each digit represents an access level: user, group, other.
- # is sum of r=4, w=2, and x=1.

Using the *numeric* method, permissions are represented by a three-digit (or four, when setting advanced permissions) *octal* number. A single octal digit can represent the numbers **0-7**, exactly the number of possibilities for a three-bit number.

To convert between symbolic and numeric representation of permissions, we need to know how the mapping is done. In the three-digit octal (numeric) representation, each digit stands for one

group of permissions, from left to right: user, group, and other. In each of these groups, start with **0**. If the read permission is present, add **4**. Add **2** if write is present, and **1** for execute.

Numeric permissions are often used by advanced administrators since they are shorter to type and pronounce, while still giving full control over all permissions.

Examine the permissions **-rwxr-x---**. For the user, **rwx** is calculated as **4+2+1=7**. For the group, **r-x** is calculated as **4+0+1=5**, and for other users, **---** is represented with **0**. Putting these three together, the numeric representation of those permissions is **750**.

This calculation can also be performed in the opposite direction. Look at the permissions **640**. For the user permissions, **6** represents read (4) and write (2), which displays as **rw-**. For the group part, **4** only includes read (4) and displays as **r--**. The **0** for other provides no permissions (**---**) and the final set of symbolic permissions for this file is **-rw-r----**.

Examples

- Remove read and write permission for group and other on **file1**:

```
[student@desktopX ~]$ chmod go-rw file1
```

- Add execute permission for everyone on **file2**:

```
[student@desktopX ~]$ chmod a+x file2
```

- Set read, write, and execute permission for user, read, and execute for group, and no permission for other on **sampledir**:

```
[student@desktopX ~]$ chmod 750 sampledir
```



Note

The **chmod** command supports the **-R** option for recursively setting permissions on an entire directory tree. When using this option, be sure to use the **X** permissions instead of the **x** permission to indicate that execute permissions should only be set on directories, and not regular files. For example, the following command will recursively set read and write access on **demodir** and all its children for their group owner, but will only apply execute permissions to directories, not regular files:

```
[student@desktopX ~]$ chmod -R g+rwx demodir
```

Changing file/directory user or group ownership

A newly created file is owned by the user who creates the file. By default, the new file has a group ownership which is the primary group of the user creating the file. Since Red Hat Enterprise Linux uses user private groups, this group is often a group with only that user as a

member. To grant access based on group membership, the owner or the group of a file may need to be changed.

File ownership can be changed with the **chown** command . For example, to grant ownership of the file **foofile** to **student**, the following command could be used:

```
[root@desktopX ~]# chown student foofile
```

chown can be used with the **-R** option to recursively change the ownership of an entire directory tree. The following command would grant ownership of **foodir** and all files and subdirectories within it to **student**:

```
[root@desktopX ~]# chown -R student foodir
```

The **chown** command can also be used to change group ownership of a file by preceding the group name with a colon (:). For example, the following command will change the group **foodir** to **admins**:

```
[root@desktopX ~]# chown :admins foodir
```

The **chown** command can also be used to change both owner and group at the same time by using the syntax **owner:group**. For example, to change the ownership of **foodir** to **visitor** and the group to **guests**, use:

```
[root@desktopX ~]# chown visitor:guests foodir
```

Only **root** can change the ownership of a file. Group ownership, however, can be set by **root** or the file's owner. **root** can grant ownership to any group, while non-**root** users can grant ownership only to groups they belong to.



Note

Instead of using **chown**, some users change the group ownership by using the **chgrp** command; this command works exactly the same as changing ownership with **chown**, including the use of **-R** to affect entire directory trees.



References

ls(1), **chmod(1)**, **chown(1)**, and **chgrp(1)** man pages

Practice: Managing File Security from the Command Line

Guided exercise

In this lab, you will create a collaborative directory for pre-existing users.

Outcomes

A directory accessible by all members of the **ateam** group and a file created by Andy that can be modified by Alice.

Before you begin...

Reset your serverX system.

- 1. Log into the GNOME desktop on serverX as **student** with a password of **password**.
- 2. Open a window with a Bash prompt.
Select Applications > Utilities > Terminal.
- 3. Become the **root** user at the shell prompt.

```
[student@serverX ~]$ su -  
Password: redhat
```

- 4. Run **lab permissions setup** which will create a shared group, **ateam**, with two new users, **andy** and **alice**. The password for these accounts is **password**

```
[root@serverX ~]# lab permissions setup
```

- 5. Create a directory in **/home** called **ateam-text**.

```
[root@serverX ~]# mkdir /home/ateam-text
```

- 6. Change the group ownership of the **ateam-text** directory to **ateam**.

```
[root@serverX ~]# chown :ateam /home/ateam-text
```

- 7. Ensure the permissions of **ateam-text** allows group members to create and delete files.

```
[root@serverX ~]# chmod g+w /home/ateam-text
```

- 8. Ensure the permissions of **ateam-text** forbids others from accessing its files.

```
[root@serverX ~]# chmod 770 /home/ateam-text  
[root@serverX ~]$ ls -ld /home/ateam-text  
drwxrwx---. 1 root ateam 6 Jan 23 12:50 /home/ateam-text
```

9. Exit the root shell and switch to the user **andy** with a password of **password**.

```
[root@serverX ~]# exit  
[student@serverX ~]$ su - andy  
Password: password
```

10. Navigate to the **/home/ateam-text** folder (remember to open a terminal window first).

```
[andy@serverX ~]$ cd /home/ateam-text
```

11. Create an empty file called **andyfile3**.

```
[andy@serverX ateam-text]$ touch andyfile3
```

12. Record the default user and group ownership of the new file and its permissions.

```
[andy@serverX ateam-text]$ ls -l andyfile3  
-rw-rw-r--. 1 andy andy 0 Jan 23 12:59 andyfile3
```

13. Change the group ownership of the new file to **ateam** and record the new ownership and permissions.

```
[andy@serverX ateam-text]$ chown :ateam andyfile3  
[andy@serverX ateam-text]$ ls -l andyfile3  
-rw-rw-r--. 1 andy ateam 0 Jan 23 12:59 andyfile3
```

14. Exit the shell and switch to the user **alice** with a password of **password**.

```
[root@serverX ~]# exit  
[student@serverX ~]$ su - alice  
Password: password
```

15. Navigate to the **/home/ateam-text** folder.

```
[alice@serverX ~]$ cd /home/ateam-text
```

16. Determine **alice**'s privileges to access and/or modify **andyfile3**.

```
[alice@serverX ateam-text]$ echo "text" >> andyfile3  
[alice@serverX ateam-text]$ cat andyfile3  
text
```

Managing Default Permissions and File Access

Objectives

After completing this section, students should be able to configure a directory in which newly created files are automatically writable by members of the group which owns the directory, using special permissions and default umask settings.

Special permissions

The **setuid** (or **setgid**) permission on an executable file means that the command will run as the **user** (or **group**) of the file, not as the user that ran the command. One example is the **passwd** command:

```
[student@desktopX ~]$ ls -l /usr/bin/passwd
-rwsr-xr-x. 1 root root 35504 Jul 16 2010 /usr/bin/passwd
```

The **sticky bit** for a directory sets a special restriction on deletion of files: Only the owner of the file (and **root**) can delete files within the directory. An example is **/tmp**:

```
[student@desktopX ~]$ ls -ld /tmp
drwxrwxrwt. 39 root root 4096 Feb 8 20:52 /tmp
```

Lastly, **setgid** on a directory means that files created in the directory will inherit the group affiliation from the directory, rather than inheriting it from the creating user. This is commonly used on group collaborative directories to automatically change a file from the default private group to the shared group.

Effects of special permissions on files and directories

Special permission	Effect on files	Effect on directories
u+s (suid)	File executes as the user that owns the file, not the user that ran the file.	No effect.
g+s (sgid)	File executes as the group that owns the file.	Files newly created in the directory have their group owner set to match the group owner of the directory.
o+t (sticky)	No effect.	Users with write on the directory can only remove files that they own; they cannot remove or force saves to files owned by other users.

Setting special permissions

- Symbolically: setuid = **u+s**; setgid = **g+s**; sticky = **o+t**
- Numerically (fourth preceding digit): setuid = 4; setgid = 2; sticky = 1

Examples

- Add the setgid bit on **directory**:

```
[root@desktopX ~]# chmod g+s directory
```

- Set the setgid bit, and read/write/execute for user and group on **directory**:

```
[root@desktopX ~]# chmod 2770 directory
```

Default file permissions

The default permissions for files are set by the processes that create them. For example, text editors create files so they are readable and writeable, but not executable, by everyone. The same goes for shell redirection. Additionally, binary executables are created executable by the compilers that create them. The **mkdir** command creates new directories with all permissions set—read, write, and execute.

Experience shows that these permissions are not typically set when new files and directories are created. This is because some of the permissions are cleared by the umask of the shell process. The **umask** command without arguments will display the current value of the shell's umask:

```
[student@desktopX ~]$ umask  
0002
```

Every process on the system has a umask, which is an octal bitmask that is used to clear the permissions of new files and directories that are created by the process. If a bit is set in the umask, then the corresponding permission is cleared in new files. For example, the previous umask, 0002, clears the write bit for other users. The leading zeros indicate the special, user, and group permissions are not cleared. A umask of 077 clears all the group and other permissions of newly created files.

Use the **umask** command with a single numeric argument to change the umask of the current shell. The numeric argument should be an octal value corresponding to the new umask value. If it is less than 3 digits, leading zeros are assumed.

The system default umask values for Bash shell users are defined in the **/etc/profile** and **/etc/bashrc** files. Users can override the system defaults in their **.bash_profile** and **.bashrc** files.

In this example, please follow along with the next steps while your instructor demonstrates the effects of **umask** on new files and directories.

1. Create a new file and directory to see how the default umask affects permissions.

```
[student@desktopX ~]$ touch newfile1  
[student@desktopX ~]$ ls -l newfile1  
-rw-rw-r--. 1 student student 0 May  9 01:54 newfile1  
[student@desktopX ~]$ mkdir newdir1  
[student@desktopX ~]$ ls -ld newdir1  
drwxrwxr-x. 2 student student 0 May  9 01:54 newdir1
```

2. Set the umask value to 0. This setting will not mask any of the permissions of new files. Create a new file and directory to see how this new umask affects permissions.

```
[student@desktopX ~]$ umask 0
[student@desktopX ~]$ touch newfile2
[student@desktopX ~]$ ls -l newfile2
-rw-rw-rw-. 1 student student 0 May  9 01:54 newfile2
[student@desktopX ~]$ mkdir newdir2
[student@desktopX ~]$ ls -ld newdir2
drwxrwxrwx. 2 student student 0 May  9 01:54 newdir2
```

3. Set the umask value to 007. This setting will mask all of the "other" permissions of new files.

```
[student@desktopX ~]$ umask 007
[student@desktopX ~]$ touch newfile3
[student@desktopX ~]$ ls -l newfile3
-rw-r----. 1 student student 0 May  9 01:55 newfile3
[student@desktopX ~]$ mkdir newdir3
[student@desktopX ~]$ ls -ld newdir3
drwxrwx---. 2 student student 0 May  9 01:54 newdir3
```

4. Set the umask value to 027. This setting will mask write access for group members and all of the "other" permissions of new files.

```
[student@desktopX ~]$ umask 027
[student@desktopX ~]$ touch newfile4
[student@desktopX ~]$ ls -l newfile4
-rw-r----. 1 student student 0 May  9 01:55 newfile4
[student@desktopX ~]$ mkdir newdir4
[student@desktopX ~]$ ls -ld newdir4
drwxr-x---. 2 student student 0 May  9 01:54 newdir4
```

5. Log in as **root** to change the default umask for unprivileged users to prohibit all access for users not in their group.

Modify **/etc/bashrc** and **/etc/profile** to change the default umask for Bash shell users. Since the default umask for unprivileged users is 0002, look for the **umask** command in these files that sets the umask to that value. Change them to set the umask to 007.

```
[root@desktopX ~]# less /etc/bashrc
# You could check uid/gid reservation validity in
# /usr/share/doc/setup-*/uidgid file
if [ $UID -gt 199 ] && [ "`id -gn`" = "`id -un`" ]; then
    umask 002
else
    umask 022
fi

# Only display echos from profile.d scripts if we are no login shell
[root@desktopX ~]# vim /etc/bashrc
[root@desktopX ~]# less /etc/bashrc
# You could check uid/gid reservation validity in
# /usr/share/doc/setup-*/uidgid file
if [ $UID -gt 199 ] && [ "`id -gn`" = "`id -un`" ]; then
    umask 007
else
```

```
        umask 022
fi

# Only display echos from profile.d scripts if we are no login shell
[root@desktopX ~]# less /etc/profile
# You could check uidgid reservation validity in
# /usr/share/doc/setup-*/uidgid file
if [ $UID -gt 199 ] && [ "`id -gn`" = "`id -un`" ]; then
    umask 002
else
    umask 022
fi

for i in /etc/profile.d/*.sh ; do
[root@desktopX ~]# vim /etc/profile
[root@desktopX ~]# less /etc/profile
# You could check uidgid reservation validity in
# /usr/share/doc/setup-*/uidgid file
if [ $UID -gt 199 ] && [ "`id -gn`" = "`id -un`" ]; then
    umask 007
else
    umask 022
fi

for i in /etc/profile.d/*.sh ; do
```

6. Log back in as **student** and confirm that the umask changes you made are persistent.

```
[student@desktopX ~]$ umask
0007
```



Note

Other shells, such as **tcsh**, may have different system default initialization files in **/etc** and users' home directories.



References

bash(1), **ls(1)**, **chmod(1)**, and **umask(1)** man pages

Practice: Controlling New File Permissions and Ownership

Guided exercise

In this lab, you will control default permissions on new files using the **umask** command and **setgid** permission.

Outcomes

- Create a shared directory where new files are automatically owned by the group **ateam**.
- Experiment with various umask settings.
- Adjust default permissions for specific users.
- Confirm your adjustment is correct.

Before you begin...

Reset your serverX system. Run **lab permissions setup** to create the **alice** account. The password for **alice** is **password**.

1. Log in as **alice** on your **serverX** virtual machine and open a window with a Bash prompt. Use the **umask** command without arguments to display Alice's default umask value.

```
[alice@serverX ~]$ umask  
0002
```

2. Create a new directory **/tmp/shared** and a new file **/tmp/shared/defaults** to see how the default umask affects permissions.

```
[alice@serverX ~]$ mkdir /tmp/shared  
[alice@serverX ~]$ ls -ld /tmp/shared  
drwxrwxr-x. 2 alice alice 6 Jan 26 18:43 /tmp/shared  
[alice@serverX ~]$ touch /tmp/shared/defaults  
[alice@serverX ~]$ ls -l /tmp/shared/defaults  
-rw-rw-r--. 1 alice alice 0 Jan 26 18:43 /tmp/shared/defaults
```

3. Change the group ownership of **/tmp/shared** to **ateam** and record the new ownership and permissions.

```
[alice@serverX ~]$ chown :ateam /tmp/shared  
[alice@serverX ~]$ ls -ld /tmp/shared  
drwxrwxr-x. 2 alice ateam 21 Jan 26 18:43 /tmp/shared
```

4. Create a new file in **/tmp/shared** and record the ownership and permissions.

```
[alice@serverX ~]$ touch /tmp/shared/alice3  
[alice@serverX ~]$ ls -l /tmp/shared/alice3  
-rw-rw-r--. 1 alice alice 0 Jan 26 18:46 /tmp/shared/alice3
```

- 5. Ensure the permissions of **/tmp/shared** cause files created in that directory to inherit the group ownership of **ateam**.

```
[alice@serverX ~]$ chmod g+s /tmp/shared
[alice@serverX ~]$ ls -ld /tmp/shared
drwxrwsr-x. 2 alice ateam 34 Jan 26 18:46 /tmp/shared
[alice@serverX ~]$ touch /tmp/shared/alice4
[alice@serverX ~]$ ls -l /tmp/shared/alice4
-rw-rw-r--. 1 alice ateam 0 Jan 26 18:48 /tmp/shared/alice4
```

- 6. Change the umask for **alice** such that new files are created with read-only access for the group and no access for other users. Create a new file and record the ownership and permissions.

```
[alice@serverX ~]$ umask 027
[alice@serverX ~]$ touch /tmp/shared/alice5
[alice@serverX ~]$ ls -l /tmp/shared/alice5
-rw-r-----. 1 alice ateam 0 Jan 26 18:48 /tmp/shared/alice5
```

- 7. Open a new Bash shell as **alice** and view the umask.

```
[alice@serverX ~]$ umask
0002
```

- 8. Change the default umask for **alice** to prohibit all access for users not in their group.

```
[alice@serverX ~]# echo "umask 007" > ~/.bashrc
[alice@serverX ~]# cat ~/.bashrc
# .bashrc

# Source global definitions
if [ -f /etc/bashrc ]; then
    . /etc/bashrc
fi

# Uncomment the following line if you don't like systemctl's auto-paging feature:
# export SYSTEMD_PAGER=

# User specific aliases and functions
umask 007
```

- 9. Log out and back into **serverX** as **alice** and confirm that the umask changes you made are persistent.

```
[alice@serverX ~]$ umask
0007
```

Lab: Controlling Access to Files with Linux File System Permissions

Performance checklist

In this lab, you will configure a system with directories for user collaboration.

Outcomes

- A directory on serverX called **/home/stooges** where these three users can work collaboratively on files.
- Only the user and group access, create, and delete files in **/home/stooges**. Files created in this directory should automatically be assigned a group ownership of **stooges**.
- New files created by users will not be accessible outside of the group.

Before you begin...

Reset your serverX system. Log into and set up your server system.

```
[student@serverX ~]$ lab permissions setup
```

Your serverX machine has three accounts, **curly**, **larry**, and **moe**, who are members of a group called **stooges**. The password for each account is **password**.

1. Open a terminal window and become root on serverX.
2. Create the **/home/stooges** directory.
3. Change group permissions on the **/home/stooges** directory so it belongs to the **stooges** group.
4. Set permissions on the **/home/stooges** directory so it is a set GID bit directory (2), the owner (7) and group (7) have full read/write/execute permissions, and other users have no permission (0) to the directory.
5. Check that the permissions were set properly.
6. Modify the global login scripts so that normal users have a umask setting which prevents others from viewing or modifying new files and directories.
7. When you finish, open a terminal window on serverX and run **lab permissions grade** to confirm you have done everything correctly.

Solution

In this lab, you will configure a system with directories for user collaboration.

Outcomes

- A directory on serverX called **/home/stooges** where these three users can work collaboratively on files.
- Only the user and group access, create, and delete files in **/home/stooges**. Files created in this directory should automatically be assigned a group ownership of **stooges**.
- New files created by users will not be accessible outside of the group.

Before you begin...

Reset your serverX system. Log into and set up your server system.

```
[student@serverX ~]$ lab permissions setup
```

Your serverX machine has three accounts, **curly**, **larry**, and **moe**, who are members of a group called **stooges**. The password for each account is **password**.

1. Open a terminal window and become root on serverX.

```
[student@serverX ~]$ su -  
Password: redhat  
[root@serverX ~]#
```

2. Create the **/home/stooges** directory.

```
[root@serverX ~]# mkdir /home/stooges
```

3. Change group permissions on the **/home/stooges** directory so it belongs to the **stooges** group.

```
[root@serverX ~]# chown :stooges /home/stooges
```

4. Set permissions on the **/home/stooges** directory so it is a set GID bit directory (2), the owner (7) and group (7) have full read/write/execute permissions, and other users have no permission (0) to the directory.

```
[root@serverX ~]# chmod 2770 /home/stooges
```

5. Check that the permissions were set properly.

```
[root@serverX ~]# ls -ld /home/stooges  
drwxrws---. 2 root stooges 1024 Dec 9 1:38 /home/stooges
```

6. Modify the global login scripts so that normal users have a umask setting which prevents others from viewing or modifying new files and directories.

```
[root@serverX ~]# vim /etc/bashrc
[root@serverX ~]# vim /etc/profile
[root@serverX ~]# less /etc/bashrc
# You could check uidgid reservation validity in
# /usr/share/doc/setup-*/uidgid file
if [ $UID -gt 199 ] && [ "`id -gn`" = "`id -un`" ]; then
    umask 007
else
    umask 022
fi

for i in /etc/profile.d/*.sh ; do
```

7. When you finish, open a terminal window on serverX and run **lab permissions grade** to confirm you have done everything correctly.

```
[student@serverX ~]$ lab permissions grade
```

Summary

Linux File System Permissions

Interpret file and directory permissions as displayed with the **ls** command.

Managing File System Permissions from the Command Line

Modify ownership and permissions of files and directories using **chmod** and **chown**.

Managing Default Permissions and File Access

Explain how default permissions are set by the system and use **umask** and **SGID** to control automatic access to files.



CHAPTER 7

MONITORING AND MANAGING LINUX PROCESSES

Overview	
Goal	To evaluate and control processes running on a Red Hat Enterprise Linux system.
Objectives	<ul style="list-style-type: none">• List and interpret basic information about processes running on the system.• Control processes in the shell's session using bash job control.• Terminate and control processes using signals.• Monitor resource usage and system load due to process activity.
Sections	<ul style="list-style-type: none">• Processes (and Practice)• Controlling Jobs (and Practice)• Killing Processes (and Practice)• Monitoring Process Activity (and Practice)
Lab	<ul style="list-style-type: none">• Monitoring and Managing Linux Processes

Processes

Objectives

After completing this section, students should be able to:

- Define the life cycle of a process.
- Define process states.
- View and interpret process listings.

What is a process?

A *process* is a running instance of a launched, executable program. A process consists of:

- an address space of allocated memory,
- security properties including ownership credentials and privileges,
- one or more execution threads of program code, and
- the process state.

The *environment* of a process includes:

- local and global variables,
- a current scheduling context, and
- allocated system resources, such as file descriptors and network ports.

An existing (*parent*) process duplicates its own address space (**fork**) to create a new (*child*) process structure. Every new process is assigned a unique *process ID* (PID) for tracking and security. The PID and the *parent's process ID* (PPID) are elements of the new process environment. Any process may create a child process. All processes are descendants of the first system process, which is **systemd(1)** on a Red Hat Enterprise Linux 7 system.

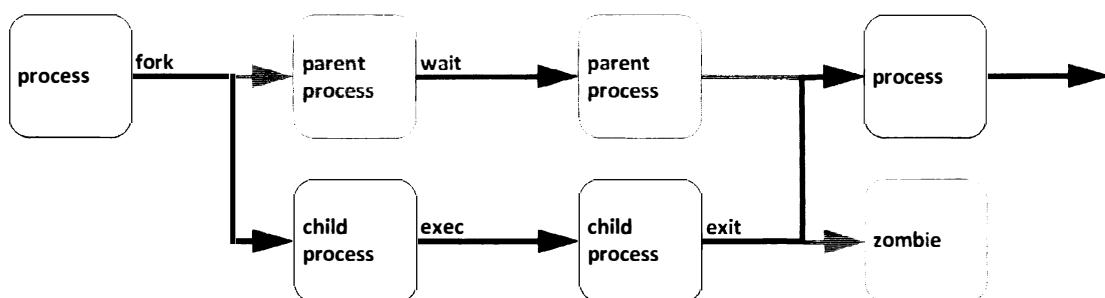


Figure 7.1: Process life cycle

Through the **fork** routine, a child process inherits security identities, previous and current file descriptors, port and resource privileges, environment variables, and program code. A child process may then **exec** its own program code. Normally, a parent process *sleeps* while the child process runs, setting a request (**wait**) to be signaled when the child completes. Upon exit, the child process has already closed or discarded its resources and environment; the remainder is

referred to as a *zombie*. The parent, signaled awake when the child exited, cleans the remaining structure, then continues with its own program code execution.

Process states

In a multitasking operating system, each CPU (or CPU core) can be working on one process at a single point in time. As a process runs, its immediate requirements for CPU time and resource allocation change. Processes are assigned a *state*, which changes as circumstances require.

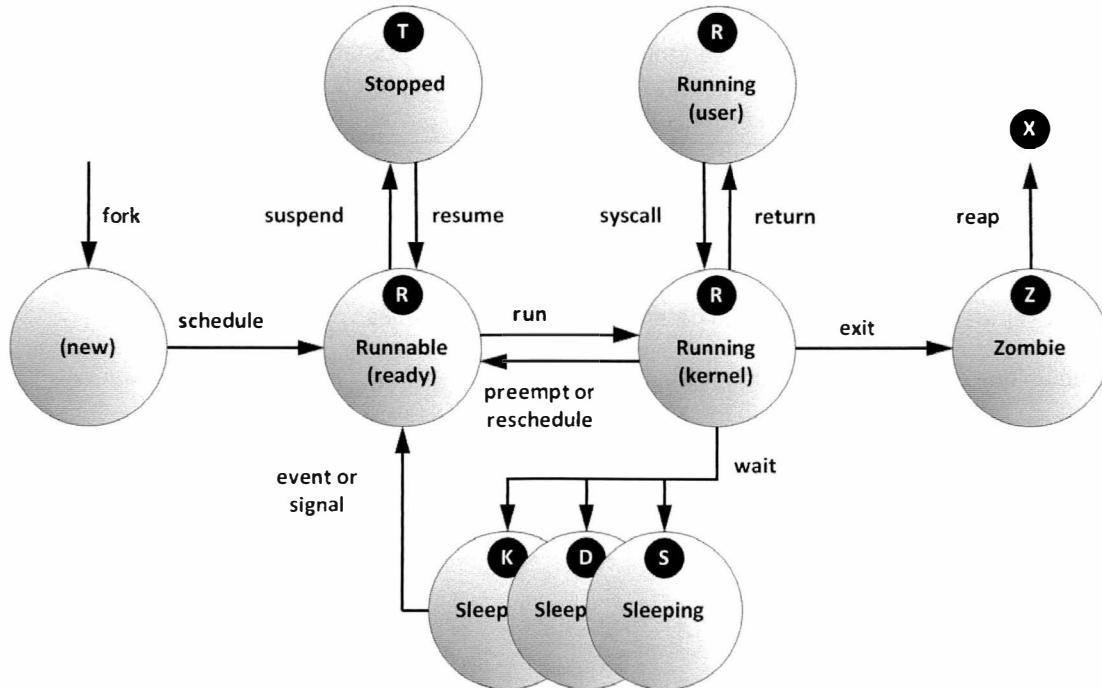


Figure 7.2: Linux process states

The Linux process states are illustrated in the previous diagram and described in the following table.

Linux process states

Name	Flag	Kernel-defined state name and description
Running	R	TASK_RUNNING: The process is either executing on a CPU or waiting to run. Process can be executing user routines or kernel routines (system calls), or be queued and ready when in the <i>Running</i> (or <i>Runnable</i>) state.
Sleeping	S	TASK_INTERRUPTIBLE: The process is waiting for some condition: a hardware request, system resource access, or signal. When an event or signal satisfies the condition, the process returns to <i>Running</i> .
	D	TASK_UNINTERRUPTIBLE: This process is also <i>Sleeping</i> , but unlike S state, will not respond to delivered signals. Used only under specific conditions in which process interruption may cause an unpredictable device state.
	K	TASK_KILLABLE: Identical to the uninterruptible D state, but modified to allow the waiting task to respond to a signal to be killed (exited completely). Utilities frequently display <i>Killable</i> processes as D state.

Name	Flag	Kernel-defined state name and description
Stopped	T	TASK_STOPPED: The process has been <i>Stopped</i> (suspended), usually by being signaled by a user or another process. The process can be continued (resumed) by another signal to return to <i>Running</i> .
	T	TASK_TRACED: A process that is being debugged is also temporarily <i>Stopped</i> and shares the same T state flag.
Zombie	Z	EXIT_ZOMBIE: A child process signals its parent as it exits. All resources except for the process identity (PID) are released.
	X	EXIT_DEAD: When the parent cleans up (<i>reaps</i>) the remaining child process structure, the process is now released completely. This state will never be observed in process-listing utilities.

List processes

The **ps** command is used for listing current processes. The command can provide detailed process information, including:

- the user identification (UID) which determines process privileges,
- the unique process identification (PID),
- the CPU and real time already expended,
- how much memory the process has allocated in various locations,
- the location of process **STDOUT**, known as the *controlling terminal*, and
- the current process state.



Important

The Linux version of **ps** supports three option formats, including:

- UNIX (POSIX) options, which may be grouped and must be preceded by a dash,
- BSD options, which may be grouped and must not be used with a dash, and
- GNU long options, which are preceded by two dashes.

For example, **ps -aux** is not the same as **ps aux**.

A common display listing (options **aux**) displays all processes, with columns in which users will be interested, and includes processes without a controlling terminal. A long listing (options **lax**) provides more technical detail, but may display faster by avoiding the username lookup. The similar UNIX syntax uses the options **-ef** to display all processes.

```
[student@serverX ~]$ ps aux
USER      PID %CPU %MEM    VSZ   RSS TTY      STAT START  TIME COMMAND
root      1  0.1  0.1  51648  7504 ?        Ss   17:45  0:03 /usr/lib/systemd/syst
root      2  0.0  0.0     0    0 ?        S    17:45  0:00 [kthreadd]
root      3  0.0  0.0     0    0 ?        S    17:45  0:00 [ksoftirqd/0]
root      5  0.0  0.0     0    0 ?        S<  17:45  0:00 [kworker/0:0H]
root      7  0.0  0.0     0    0 ?        S    17:45  0:00 [migration/0]
-- output truncated --
```

```
[student@serverX ~]$ ps lax
F  UID   PID  PPID PRI  NI    VSZ   RSS WCHAN STAT TTY      TIME COMMAND
4   0     1     0  20   0  51648  7504 ep_pol Ss   ?
0:03 /usr/lib/systemd/
1   0     2     0  20   0     0  kthrea S   ?
0:00 [kthreadd]
1   0     3     2  20   0     0  smpboo S   ?
0:00 [ksoftirqd/0]
1   0     5     2  0 -20   0  worker S<  ?
0:00 [kworker/0:0H]
1   0     7     2 -100  -   0  smpboo S   ?
0:00 [migration/0]
-- output truncated --
[student@serverX ~]$ ps -ef
UID      PID  PPID  C STIME TTY      TIME CMD
root      1     0  0 17:45 ?
00:00:03 /usr/lib/systemd/systemd --switched-ro
root      2     0  0 17:45 ?
00:00:00 [kthreadd]
root      3     2  0 17:45 ?
00:00:00 [ksoftirqd/0]
root      5     2  0 17:45 ?
00:00:00 [kworker/0:0H]
root      7     2  0 17:45 ?
00:00:00 [migration/0]
-- output truncated --
```

By default, **ps** with no options selects all processes with the same *effective user ID* (EUID) as the current user and associated with the same terminal where **ps** was invoked.

- Processes in brackets (usually at the top) are scheduled kernel threads.
- Zombies show up in a ps listing as *exiting* or *defunct*.
- **ps** displays once. Use **top(1)** for a repetitive update process display.
- **ps** can display in tree format to view parent/child relationships.
- The default output is unsorted. Display order matches that of the system process table, which reuses table rows as processes die and new ones are created. Output may appear chronological, but is not guaranteed unless explicit **-o** or **--sort** options are used.



References

info libc signal (*GNU C Library Reference Manual*)

- Section 24: Signal Handling

info libc processes (*GNU C Library Reference Manual*)

- Section 26: Processes

ps(1) and **signal(7)** man pages

Practice: Processes

Quiz

Match the following items to their counterparts in the table.



Description	State
Process has been stopped temporarily.	
Process has just been terminated.	
Process is scheduled but is not yet on a CPU.	
Process is waiting for I/O.	
Process is uninterruptibly waiting for a device to respond.	
Process is at a prompt, needing user input.	
Process is executing a system call.	

Solution

Match the following items to their counterparts in the table.

Description	State
Process has been stopped temporarily.	T
Process has just been terminated.	Z
Process is scheduled but is not yet on a CPU.	R
Process is waiting for I/O.	S
Process is uninterruptibly waiting for a device to respond.	D
Process is at a prompt, needing user input.	S
Process is executing a system call.	R

Controlling Jobs

Objectives

After completing this section, students should be able to:

- Explain the terms foreground, background, and controlling terminal.
- Use job control to manage multiple command-line tasks.

Jobs and sessions

Job control is a command shell feature allowing a single shell instance to run and manage multiple commands. Without job control, a parent shell forks a child process to run a command, sleeping until the child process exits. When the shell prompt redisplays, the parent shell has returned. With job control, commands can be selectively suspended, resumed, and run asynchronously, allowing the shell to return for additional commands while child processes run.

A *foreground* process is a command running in a terminal window. The terminal's device ID (**tty**) is the process's *controlling terminal*. Foreground processes receive keyboard-generated input and signals and are allowed to read from or write to the terminal (e.g., via **stdin** and **stdout**).

A process *session* is created when a terminal or console first opens (e.g., at login or by invoking a new **Terminal** instance). All processes (e.g., the first command shell, its children, and pipelines) initiated from that terminal share the same *session ID*. Within a session, only one process can be in the foreground at a time.

A *background* process is started *without* a controlling terminal because it has no need for terminal interaction. In a **ps** listing, such processes (e.g., service daemons and kernel process threads) display a question mark (?) in the **TTY** column. Background processes which (improperly) attempt to read from or write to the terminal may be suspended.

Running jobs in the background

Any command can be started in the background by appending an ampersand (&) to the command line. The **bash** shell displays a *job number* (unique to the session) and the PID of the new child process. The command shell does not wait for the child process and redisplays the shell prompt.

```
[student@serverX ~]$ sleep 10000 &
[1] 5947
[student@serverX ~]$
```



Note

An ampersand will background only the last command in a pipeline, unless the command set is surrounded with parentheses. The PID returned is from the pipeline's last process. All processes in the pipeline are now members of the same job.

```
[student@serverX ~]$ ( example_command | sort | mail -s "Sort output") &
[1] 5998
```

The **bash** command shell tracks jobs, per session, in a table displayed with the **jobs** command.

```
[student@serverX ~]$ jobs  
[1]+  Running                  sleep 10000 &  
[student@serverX ~]$
```

Background jobs can reconnect to the controlling terminal by being brought to the foreground using the **fg** command with the job ID (**%job number**).

```
[student@serverX ~]$ fg %1  
sleep 10000  
-
```

The example **sleep** command is now running on the controlling terminal. The command shell is again asleep, waiting for this child process to exit. To resend to the background, or to send any command in which the trailing ampersand was not originally included, send the keyboard-generated suspend request (**Ctrl-z**) to the process.

```
sleep 10000  
^Z  
[1]+  Stopped                  sleep 10000  
[student@serverX ~]$
```

The suspend takes effect immediately. The job is placed in the background. Pending output and keyboard typeahead are discarded.

The **ps** option **j** displays job information, including the initial command shell of each session. Since the **sleep** example command is currently suspended, the state flag displayed is **T**.

```
[student@serverX ~]$ ps j  
PPID  PID  PGID  SID TTY      TPGID STAT   UID    TIME COMMAND  
2764  2768  2768  2768 pts/0    6377 Ss    1000   0:00 /bin/bash  
2768  5947  5947  2768 pts/0    6377 T     1000   0:00 sleep 10000  
2768  6377  6377  2768 pts/0    6377 R+    1000   0:00 ps j  
[student@serverX ~]$
```

To restart the process in the background, use the **bg** command with the same job ID.

```
[student@serverX ~]$ bg %1  
[1]+ sleep 10000 &  
[student@serverX ~]$
```

The command shell will warn a user who attempts to exit a terminal window (session) with suspended jobs. If the user tries exiting again immediately, the suspended jobs are killed.

R

References

Additional information may be available in the chapter on viewing system processes in the *Red Hat Enterprise Linux System Administrator's Guide* for Red Hat Enterprise Linux 7, which can be found at

<http://docs.redhat.com/>

libc info page (*GNU C Library Reference Manual*)

- Section 24: Signal Handling
- Section 26: Processes

bash(1), **builtins(1)**, **ps(1)**, **sleep(1)** man pages

Practice: Background and Foreground Processes

Guided exercise

In this lab, students will start, suspend, and reconnect to multiple processes using job control.

Outcomes:

Practice suspending and restarting user processes.

Before you begin...

Log in as student to serverX. Begin in student's home directory.

- 1. Open two terminal windows, side by side, to be referred to as *left* and *right*.
- 2. In the left window, start a process that continuously appends the word "rock" and a space to the file **~/outfile** at one-second intervals. The complete command set must be contained in parentheses for job control to interpret the set as a single job.

```
[student@serverX ~]$ (while true; do echo -n "rock " >> ~/outfile; sleep 1; done)
```

- 3. In the right window, use **tail** to confirm that the new process is writing to the file.

```
[student@serverX ~]$ tail -f ~/outfile
```

- 4. In the left window, suspend the running process. The shell returns the job ID in square brackets. In the right window, confirm that the process output has stopped.

```
[student@serverX ~]$ Ctrl-z
```

- 5. In the left window, view the **jobs** list. The **+** denotes the *current job*. Restart the job in the background. In the right window, confirm that the process output is again active.

```
[student@serverX ~]$ jobs  
[1]+ Stopped ( while true; do  
    echo -n "rock " >> ~/outfile; sleep 1;  
done )  
[student@serverX ~]$ bg  
[student@serverX ~]$ jobs
```

- 6. In the left window, start two more processes to append to the same output file. Replace "rock" with "paper," and then with "scissors." To properly background the process, the complete command set must be contained in parentheses and ended with an ampersand.

```
[student@serverX ~]$ (while true; do echo -n "paper " >> ~/outfile; sleep 1;  
done) &  
[student@serverX ~]$ (while true; do echo -n "scissors " >> ~/outfile; sleep 1;  
done) &
```

- 7. In the left window, view **jobs** to see all three processes "Running". In the right window, confirm that all three processes are appending to the file.

```
[student@serverX ~]$ jobs
```

- 8. Using only commands previously learned, suspend the "rock" process. In the left window, foreground the job, using the job ID determined from the **jobs** listing, then suspend it using Ctrl-z. Confirm that the "rock" process is "Stopped". In the right window, confirm that "rock" output is no longer active.

```
[student@serverX ~]$ jobs  
[student@serverX ~]$ fg %number  
[student@serverX ~]$ Ctrl-z
```

- 9. End the "paper" process. In the left window, foreground the job, then terminate it using Ctrl-c. Confirm that the "paper" process has disappeared. In the right window, confirm that "paper" output is no longer active.

```
[student@serverX ~]$ jobs  
[student@serverX ~]$ fg %number  
[student@serverX ~]$ Ctrl-c
```

- 10. In the left window, view the remaining jobs using **ps**. The suspended job has state **T**. The other background job is sleeping (**S**), since **ps** is "on cpu" (**R**) while displaying.

```
[student@serverX ~]$ ps j  
  PID  PID  PGID  SID TTY      TPGID STAT   UID    TIME COMMAND  
4489  6223  6223  6223 pts/1    12918 Ss    1000   0:00 bash  
4489  6237  6237  6237 pts/2    9782 Ss    1000   0:00 bash  
6237  9782  9782  6237 pts/2    9782 S+    1000   0:00 tail -f /home/student/o  
7360  9856  7360  6223 pts/1    12918 T     1000   0:00 sleep 1  
7395  12916 7395  6223 pts/1    12918 S     1000   0:00 sleep 1  
6223  12918 12918 6223 pts/1    12918 R+    1000   0:00 ps j
```

- 11. Stop the remaining two jobs. In the left window, foreground either job. Terminate it using Ctrl-c. Repeat with the remaining job. The "Stopped" job temporarily restarts when foregrounded. Confirm that no jobs remain and that output has stopped.

```
[student@serverX ~]$ fg %number  
[student@serverX ~]$ Ctrl-c  
[student@serverX ~]$ fg %number  
[student@serverX ~]$ Ctrl-c  
[student@serverX ~]$ jobs
```

- 12. In the right window, stop the **tail** command. Close extra terminal windows.

```
[student@serverX ~]$ Ctrl-c
```

Killing Processes

Objectives

After completing this section, students should be able to:

- Use commands to kill and communicate with processes.
- Define the characteristics of a daemon process.
- End user sessions and processes.

Process control using signals

A signal is a software interrupt delivered to a process. Signals report events to an executing program. Events that generate a signal can be an *error*, *external event* (e.g., I/O request or expired timer), or by *explicit request* (e.g., use of a signal-sending command or by keyboard sequence).

The following table lists the fundamental signals used by system administrators for routine process management. Refer to signals by either their short (**HUP**) or proper (**SIGHUP**) name.

Fundamental process management signals

Signal number	Short name	Definition	Purpose
1	HUP	Hangup	Used to report termination of the controlling process of a terminal. Also used to request process reinitialization (configuration reload) without termination.
2	INT	Keyboard interrupt	Causes program termination. Can be blocked or handled. Sent by typing INTR character (Ctrl-c).
3	QUIT	Keyboard quit	Similar to SIGINT , but also produces a process dump at termination. Sent by typing QUIT character (Ctrl-\).
9	KILL	Kill, unblockable	Causes abrupt program termination. Cannot be blocked, ignored, or handled; always fatal.
15 <i>default</i>	TERM	Terminate	Causes program termination. Unlike SIGKILL , can be blocked, ignored, or handled. The polite way to ask a program to terminate; allows self-cleanup.
18	CONT	Continue	Sent to a process to resume if stopped. Cannot be blocked. Even if handled, always resumes the process.
19	STOP	Stop, unblockable	Suspends the process. Cannot be blocked or handled.
20	TSTP	Keyboard stop	Unlike SIGSTOP , can be blocked, ignored, or handled. Sent by typing SUSP character (Ctrl-z).



Note

Signal numbers vary on different Linux hardware platforms, but signal names and meanings are standardized. For command use, it is advised to use signal names instead of numbers. The numbers discussed in this section are for Intel x86 systems.

Each signal has a *default action*, usually one of the following:

Term – Cause a program to terminate (exit) at once.

Core – Cause a program to save a memory image (core dump), then terminate.

Stop – Cause a program to stop executing (suspend) and wait to continue (resume).

Programs can be prepared for expected event signals by implementing handler routines to ignore, replace, or extend a signal's default action.

Commands for sending signals by explicit request

Users signal their current foreground process by typing a keyboard control sequence to suspend (**Ctrl-z**), kill (**Ctrl-c**), or core dump (**Ctrl-**) the process. To signal a background process or processes in a different session requires a signal-sending command.

Signals can be specified either by name (e.g., **-HUP** or **-SIGHUP**) or by number (e.g., **-1**). Users may kill their own processes, but root privilege is required to kill processes owned by others.

- The **kill** command sends a signal to a process by ID. Despite its name, the **kill** command can be used for sending any signal, not just those for terminating programs.

```
[student@serverX ~]$ kill PID
[student@serverX ~]$ kill -signal PID
[student@serverX ~]$ kill -1
 1) SIGHUP      2) SIGINT      3) SIGQUIT      4) SIGILL      5) SIGTRAP
 6) SIGABRT     7) SIGBUS      8) SIGFPE       9) SIGKILL     10) SIGUSR1
11) SIGSEGV     12) SIGUSR2     13) SIGPIPE     14) SIGALRM     15) SIGTERM
16) SIGSTKFLT   17) SIGCHLD     18) SIGCONT     19) SIGSTOP     20) SIGTSTP
-- output truncated --
```

- Use **killall** to send a signal to one or more processes matching selection criteria, such as a command name, processes owned by a specific user, or all system-wide processes.

```
[student@serverX ~]$ killall command_pattern
[student@serverX ~]$ killall -signal command_pattern
[root@serverX ~]# killall -signal -u username command_pattern
```

- The **pkill** command, like **killall**, can signal multiple processes. **pkill** uses advanced selection criteria, which can include combinations of:

Command – Processes with a pattern-matched command name.

UID – Processes owned by a Linux user account, effective or real.

GID – Processes owned by a Linux group account, effective or real.

Parent – Child processes of a specific parent process.

Terminal – Processes running on a specific controlling terminal.

```
[student@serverX ~]$ pkill command_pattern
[student@serverX ~]$ pkill -signal command_pattern
[root@serverX ~]# pkill -G GID command_pattern
```

```
[root@serverX ~]# pkill -P PPID command_pattern
[root@serverX ~]# pkill -t terminal_name -U UID command_pattern
```

Logging users out administratively

The **w** command views users currently logged into the system and their cumulative activities. Use the **TTY** and **FROM** columns to determine the user's location.

All users have a controlling terminal, listed as **pts/N** while working in a graphical environment window (*pseudo-terminal*) or **ttyN** on a system console, alternate console, or other directly connected terminal device. Remote users display their connecting system name in the **FROM** column when using the **-f** option.

```
[student@serverX ~]$ w -f
12:43:06 up 27 min,  5 users,  load average: 0.03,  0.17,  0.66
USER    TTY     FROM          LOGIN@   IDLE    JCPU   PCPU WHAT
student :0      :0            12:20   ?xdm?   1:10   0.16s gdm-session-wor
student pts/0   :0            12:20   2.00s  0.08s  0.01s w -f
root    tty2          :0            12:26   14:58   0.04s  0.04s -bash
bob     tty3          :0            12:28   14:42   0.02s  0.02s -bash
student pts/1   desktop2.example.12:41  1:07   0.03s  0.03s -bash
[student@serverX ~]$
```

Discover how long a user has been on the system by viewing the session login time. For each session, CPU resources consumed by current jobs, included background tasks and children , are in the **JCPU** column. Current foreground process CPU consumption is in the **PCPU** column.

Users may be forced off a system for security violations, resource overallocation, or other administrative need. Users are expected to quit unnecessary applications, close unused command shells, and exit login sessions when requested.

When situations occur in which users cannot be contacted or have unresponsive sessions, runaway resource consumption, or improper system access, their sessions may need to be administratively terminated using signals.



Important

Although **SIGTERM** is the default signal, **SIGKILL** is a commonly misused administrator favorite. Since the **SIGKILL** signal cannot be handled or ignored, it is always fatal. However, it forces termination without allowing the killed process to run self-cleanup routines. It is recommended to send **SIGTERM** first, then retry with **SIGKILL** only if a process fails to respond.

Processes and sessions can be individually or collectively signaled. To terminate all processes for one user, use the **pkill** command. Because the initial process in a login session (*session leader*) is designed to handle session termination requests and ignore unintended keyboard signals, killing all of a user's processes and login shells requires using the **SIGKILL** signal.

```
[root@serverX ~]# pgrep -l -u bob
6964 bash
6998 sleep
6999 sleep
7000 sleep
[root@serverX ~]# pkill -SIGKILL -u bob
```

```
[root@serverX ~]# pgrep -l -u bob
[root@serverX ~]#
```

When processes requiring attention are in the same login session, it may not be necessary to kill all of a user's processes. Determine the controlling terminal for the session using the **w** command, then kill only processes which reference the same terminal ID. Unless **SIGKILL** is specified, the session leader (here, the **bash** login shell) successfully handles and survives the termination request, but all other session processes are terminated.

```
[root@serverX ~]# pgrep -l -u bob
7391 bash
7426 sleep
7427 sleep
7428 sleep
[root@serverX ~]# w -h -u bob
bob      tty3        18:37   5:04   0.03s  0.03s -bash
[root@serverX ~]# pkill -t tty3
[root@serverX ~]# pgrep -l -u bob
7391 bash
[root@serverX ~]# pkill -SIGKILL -t tty3
[root@serverX ~]# pgrep -l -u bob
[root@serverX ~]#
```

The same selective process termination can be applied using parent and child process relationships. Use the **pstree** command to view a process tree for the system or a single user. Use the parent process's PID to kill all children they have created. This time, the parent **bash** login shell survives because the signal is directed only at its child processes.

```
[root@serverX ~]# pstree -p bob
bash(8391)─sleep(8425)
             ├sleep(8426)
             └sleep(8427)
[root@serverX ~]# pkill -P 8391
[root@serverX ~]# pgrep -l -u bob
bash(8391)
[root@serverX ~]# pkill -SIGKILL -P 8391
[root@serverX ~]# pgrep -l -u bob
bash(8391)
[root@serverX ~]#
```

References



info libc signal (*GNU C Library Reference Manual*)

- Section 24: Signal Handling

info libc processes (*GNU C Library Reference Manual*)

- Section 26: Processes

kill(1), **killall(1)**, **pgrep(1)**, **pkill(1)**, **pstree(1)**, **signal(7)**, and **w(1)** man pages

Practice: Killing Processes

Guided exercise

In this lab, students will use keyboard sequences and signals to manage and stop processes.

Outcomes:

Experience with observing the results of starting and stopping multiple shell processes.

Before you begin...

Log in as student to serverX. Start in your home directory.

- 1. Open two terminal windows, side by side, to be referred to as *left* and *right*.
- 2. In the left window, start three processes that append text to an output file at one-second intervals. To properly background each process, the complete command set must be contained in parentheses and ended with an ampersand.

```
[student@serverX ~]$ (while true; do echo -n "game " >> ~/outfile; sleep 1; done)
&
[student@serverX ~]$ (while true; do echo -n "set " >> ~/outfile; sleep 1; done)
&
[student@serverX ~]$ (while true; do echo -n "match " >> ~/outfile; sleep 1;
done) &
```

- 3. In the right window, use **tail** to confirm that all three processes are appending to the file. In the left window, view **jobs** to see all three processes "Running".

```
[student@serverX ~]$ tail -f ~/outfile
[student@serverX ~]$ jobs
[1]  Running                  ( while true; do
    echo -n "game " >> ~/outfile; sleep 1;
done ) &
[2]- Running                  ( while true; do
    echo -n "set " >> ~/outfile; sleep 1;
done ) &
[3]+ Running                  ( while true; do
    echo -n "match " >> ~/outfile; sleep 1;
done ) &
```

- 4. Suspend the "game" process using signals. Confirm that the "game" process is "Stopped". In the right window, confirm that "game" output is no longer active.

```
[student@serverX ~]$ kill -SIGSTOP %number
[student@serverX ~]$ jobs
```

- 5. Terminate the "set" process using signals. Confirm that the "set" process has disappeared. In the right window, confirm that "set" output is no longer active.

```
[student@serverX ~]$ kill -SIGTERM %number
[student@serverX ~]$ jobs
```

- 6. Resume the "game" process using signals. Confirm that the "game" process is "Running". In the right window, confirm that "game" output is again active.

```
[student@serverX ~]$ kill -SIGCONT %number  
[student@serverX ~]$ jobs
```

- 7. Terminate the remaining two jobs. Confirm that no jobs remain and that output has stopped. From the left window, terminate the right window's **tail** command.

Close extra terminal windows.

```
[student@serverX ~]$ kill -SIGTERM %number  
[student@serverX ~]$ kill -SIGTERM %number  
[student@serverX ~]$ jobs  
[student@serverX ~]$ pkill -SIGTERM tail  
[student@serverX ~]$
```

Monitoring Process Activity

Objectives

After completing this section, students should be able to:

- Interpret uptime and load averages.
- Monitor real-time processes.

Load average

The Linux kernel calculates a *load average* metric as an *exponential moving average* of the *load number*, a cumulative CPU count of active system resource requests.

- *Active requests* are counted from per-CPU queues for running threads and threads waiting for I/O, as the kernel tracks process resource activity and corresponding process state changes.
- *Load number* is a calculation routine run every five seconds by default, which accumulates and averages the active requests into a single number for all CPUs.
- *Exponential moving average* is a mathematical formula to smooth out trending data highs and lows, increase current activity significance, and decrease aging data quality.
- *Load average* is the load number calculation routine result. Collectively, it refers to the three displayed values of system activity data averaged for the last 1, 5, and 15 minutes.

Understanding the Linux load average calculation

The load average represents the perceived system load over a time period. Linux implements the load average calculation as a representation of expected service wait times, not only for CPU but also for disk and network I/O.

- Linux counts not only processes, but threads individually, as separate tasks. CPU request queues for running threads (*nr_running*) and threads waiting for I/O resources (*nr_iowait*) reasonably correspond to process states **R** (*Running*) and **D** (*Uninterruptable Sleeping*). Waiting for I/O includes tasks sleeping for expected disk and network responses.
- The load number is a global counter calculation, which is sum-totaled for all CPUs. Since tasks returning from sleep may reschedule to different CPUs, accurate per-CPU counts are difficult, but an accurate cumulative count is assured. Displayed load averages represent all CPUs.
- Linux counts each physical CPU core and microprocessor hyperthread as separate execution units, logically represented and referred to as individual CPUs. Each CPU has independent request queues. View **/proc/cpuinfo** for the kernel representation of system CPUs.

```
[student@serverX ~]$ grep "model name" /proc/cpuinfo
model name : Intel(R) Core(TM) i5 CPU      M 520 @ 2.40GHz
model name : Intel(R) Core(TM) i5 CPU      M 520 @ 2.40GHz
model name : Intel(R) Core(TM) i5 CPU      M 520 @ 2.40GHz
model name : Intel(R) Core(TM) i5 CPU      M 520 @ 2.40GHz
[student@serverX ~]$ grep "model name" /proc/cpuinfo | wc -l
4
```

- Some UNIX systems only considered CPU utilization or run queue length to indicate system load. Since a system with idle CPUs can experience extensive waiting due to busy disk or network resources, I/O consideration is included in the Linux load average. When experiencing high load averages with minimal CPU activity, examine the disk and network activity.

Interpreting displayed load average values

The three values represent the weighted values over the last 1, 5, and 15 minutes. A quick glance can indicate whether system load appears to be increasing or decreasing. Calculate the approximate *per-CPU* load value to determine whether the system is experiencing significant waiting.

- top**, **uptime**, **w**, and **gnome-system-monitor** display load average values.

```
[student@serverX ~]$ uptime  
15:29:03 up 14 min, 2 users, load average: 2.92, 4.48, 5.20
```

- Divide the displayed load average values by the number of logical CPUs in the system. A value below 1 indicates satisfactory resource utilization and minimal wait times. A value above 1 indicates resource saturation and some amount of service waiting times.

```
# From /proc/cpuinfo, system has four logical CPUs, so divide by 4:  
#           load average: 2.92, 4.48, 5.20  
#           divide by number of logical CPUs:   4   4   4  
#                                         -----  
#           per-CPU load average: 0.73  1.12  1.30  
#  
# This system's load average appears to be decreasing.  
# With a load average of 2.92 on four CPUs, all CPUs were in use ~73% of the time.  
# During the last 5 minutes, the system was overloaded by ~12%.  
# During the last 15 minutes, the system was overloaded by ~30%.
```

- An idle CPU queue has a load number of 0. Each ready and waiting thread adds a count of 1. With a total queue count of 1, the resource (CPU, disk, or network) is in use, but no requests spend time waiting. Additional requests increment the count, but since many requests can be processed within the time period, resource *utilization* increases, but not *wait times*.
- Processes sleeping for I/O due to a busy disk or network resource are included in the count and increase the load average. While not an indication of CPU utilization, the queue count still indicates that users and programs are waiting for resource services.
- Until resource saturation, a load average will remain below 1, since tasks will seldom be found waiting in queue. Load average only increases when resource saturation causes requests to remain queued and counted by the load calculation routine. When resource utilization approaches 100%, each additional request starts experiencing service wait time.

Real-time process monitoring

The **top** program is a dynamic view of the system's processes, displaying a summary header followed by a process or thread list similar to **ps** information. Unlike the static **ps** output, **top** continuously refreshes at a configurable interval, and provides capabilities for column reordering, sorting, and highlighting. User configurations can be saved and made persistent.

Default output columns are recognizable from other resource tools:

- The process ID (**PID**).

- User name (**USER**) is the process owner.
- Virtual memory (**VIRT**) is all memory the process is using, including the resident set, shared libraries, and any mapped or swapped memory pages. (Labeled **VSZ** in the **ps** command.)
- Resident memory (**RES**) is the physical memory used by the process, including any resident shared objects. (Labeled **RSS** in the **ps** command.)
- Process state (**S**) displays as:
 - **D** = Uninterruptable Sleeping
 - **R** = Running or Runnable
 - **S** = Sleeping
 - **T** = Stopped or Traced
 - **Z** = Zombie
- CPU time (**TIME**) is the total processing time since the process started. May be toggled to include cumulative time of all previous children.
- The process command name (**COMMAND**).

Fundamental keystrokes in top

Key	Purpose
? or h	Help for interactive keystrokes.
l, t, m	Toggles for load, threads, and memory header lines.
1	Toggle showing individual CPUs or a summary for all CPUs in header.
s⁽¹⁾	Change the refresh (screen) rate, in decimal seconds (e.g., 0.5, 1, 5).
b	Toggle reverse highlighting for <i>Running</i> processes; default is bold only.
B	Enables use of bold in display, in the header, and for <i>Running</i> processes.
H	Toggle threads; show process summary or individual threads.
u, U	Filter for any user name (effective, real).
M	Sorts process listing by memory usage, in descending order.
P	Sorts process listing by processor utilization, in descending order.
k⁽¹⁾	Kill a process. When prompted, enter PID , then signal .
r⁽¹⁾	Renice a process. When prompted, enter PID , then nice_value .
w	Write (save) the current display configuration for use at the next top restart.
q	Quit.
Note: ⁽¹⁾ Not available if top started in secure mode. See top(1) .	



References

GNOME System Monitor

- **yelp help:gnome-system-monitor**

ps(1), **top(1)**, **uptime(1)**, and **w(1)** man pages

Practice: Monitoring Process Activity

Guided exercise

In this lab, students will use the **top** command to dynamically view, sort, and stop processes.

Outcomes

Practice with managing processes in real time.

Before you begin...

Perform the following tasks as **student** on the serverX machine. Run **lab process101 setup** on serverX to prepare for this exercise.

```
[student@serverX ~]$ lab process101 setup
```

- 1. Open two terminal windows, side by side, to be referred to as *left* and *right*. In the right terminal, run the **top** utility. Size the window to be as tall as possible.

```
[student@serverX ~]$ top
```

- 2. In the left terminal, determine the number of logical CPUs on this virtual machine.

```
[student@serverX ~]$ grep "model name" /proc/cpuinfo | wc -l  
1
```

- 3. In the left terminal, run a single instance of the **process101** executable.

```
[student@serverX ~]$ process101
```

- 4. In the right terminal, observe the **top** display. Use the single keystrokes **l**, **t**, and **m** to toggle the load, threads, and memory header lines. After observing this behavior, ensure that all headers are displaying.

- 5. Note the process ID (PID) for **process101**. View the CPU percentage for the process, which is expected to hover around 25% or 30%.

View the load averages. On a single-CPU virtual machine, for example, the one-minute load average is currently less than a value of 1. The value observed may be affected by resource contention from another virtual machine or the virtual host.

- 6. In the left terminal, run a second instance of **process101**.

```
[student@serverX ~]$ process101
```

- 7. In **top**, note the process ID (PID) for the second **process101**. View the CPU percentage for the process, also expected to hover around 25% or 30%.

View the one-minute load average again, which may still be less than 1. Wait up to one minute to allow the calculation to adjust to the new workload.

- 8. In the left terminal, run a third instance of **process101**.

```
[student@serverX ~]$ process101
```

- 9. In **top**, note the process ID (PID) for the third **process101**. View the CPU percentage for the process, again expected to hover around 25% or 30%.

View the one-minute load average again, which now is expected to be above 1. Wait up to one minute to allow the calculation to again adjust to the new workload.

- 10. *Optional:* If this virtual machine has more than one logical CPU, slowly start additional **process101** instances until the one-minute load average equals or exceeds the number of logical CPUs. Divide the load average value by the number of CPUs to determine the estimated load average per CPU.
- 11. When finished observing the load average values, terminate each of the **process101** processes from within **top**.

- 11.1. Press **k**. Observe the prompt below the headers and above the columns.

- 11.2. Type the PID for one of the **process101** instances. Press **Enter**.

- 11.3. Press **Enter** again to use the default **SIGTERM** signal **15**.

Confirm that the selected process is no longer observed in **top**. If the PID still remains, repeat these terminating steps, substituting **SIGKILL** signal **9** when prompted.

- 12. Repeat the previous step for each remaining **process101** instance. Confirm that no **process101** instances remain in **top**.
- 13. In the right window, press **q** to exit **top**. Close extra terminal windows.

Lab: Monitoring and Managing Linux Processes

Performance checklist

In this lab, students will locate and manage processes that are using the most resources on a system.

Outcomes

Experience using **top** as a process management tool.

Before you begin...

Perform the following tasks as **student** on the serverX machine. Run **lab processes setup** as **root** on serverX to prepare for this exercise.

```
[student@serverX ~]$ lab processes setup
```

1. In a terminal window, run the **top** utility. Size the window to be as tall as possible.
2. Observe the **top** display. The default display sorts by CPU utilization, highest first. What are the processes using the most CPU time?
3. Change the display to sort by the amount of memory in use by each process.
4. What are the processes with the largest memory allocations?
5. Turn off the use of bold in the display. Save this configuration for reuse when **top** is restarted.
6. Exit **top**, then restart it again. Confirm that the new display uses the saved configuration; i.e., the display starts sorted by memory utilization and bold is turned off.
7. Modify the display to again sort by CPU utilization. Turn on the use of bold. Observe that only *Running* or *Runnable* (state **R**) process entries are bold. Save this configuration.
8. Open another terminal window if necessary. As **root**, suspend the **hippo** process. In **top**, observe that the process state is now **T**.
9. The **hippo** process quickly disappears from the display, since it is no longer actively using CPU resources. List the process information from the command line to confirm the process state.
10. Resume execution of the **hippo** processes.
11. When finished observing the display, terminate the extra processes using the command line. Confirm that the processes no longer display in **top**.
12. Check that the cleanup is successful by running the grading script. If necessary, find and terminate processes listed by the grading script, and repeat grading.
13. Exit the **top** display. Close extra terminal windows.

Solution

In this lab, students will locate and manage processes that are using the most resources on a system.

Outcomes

Experience using **top** as a process management tool.

Before you begin...

Perform the following tasks as **student** on the serverX machine. Run **lab processes setup** as **root** on serverX to prepare for this exercise.

```
[student@serverX ~]$ lab processes setup
```

1. In a terminal window, run the **top** utility. Size the window to be as tall as possible.

```
[student@serverX ~]$ top
top - 12:47:46 up 2:02, 3 users, load average: 1.67, 1.25, 0.73
Tasks: 361 total, 6 running, 355 sleeping, 0 stopped, 0 zombie
%Cpu(s): 98.5 us, 1.4 sy, 0.0 ni, 0.0 id, 0.0 wa, 0.0 hi, 0.1 si, 0.0 st
KiB Mem: 2043424 total, 897112 used, 1146312 free, 1740 buffers
KiB Swap: 4079612 total, 0 used, 4079612 free. 296276 cached Me

      PID USER      PR  NI      VIRT      RES      SHR S %CPU %MEM     TIME+ COMMAND
  4019 root      20   0    4156       76      0 R 57.5  0.0  2:54.15 hippo
 2492 student   20   0 1359500 168420  37492 S 16.8  8.2  3:55.58 gnome-shell
 1938 root      20   0 189648  35972    7568 R  1.9  1.8  0:29.66 Xorg
 2761 student   20   0  620192  19688   12296 S  0.4  1.0  0:04.48 gnome-terminal
output truncated
```

2. Observe the **top** display. The default display sorts by CPU utilization, highest first. What are the processes using the most CPU time?

In addition to the default GNOME shell, find the process named **hippo**.

3. Change the display to sort by the amount of memory in use by each process.

Press **M**.

```
top - 12:57:38 up 2:11, 3 users, load average: 2.09, 1.70, 1.19
Tasks: 360 total, 5 running, 355 sleeping, 0 stopped, 0 zombie
%Cpu(s): 99.8 us, 0.2 sy, 0.0 ni, 0.0 id, 0.0 wa, 0.0 hi, 0.0 si, 0.0 st
KiB Mem: 2043424 total, 896952 used, 1146472 free, 1740 buffers
KiB Swap: 4079612 total, 0 used, 4079612 free. 296280 cached Mem

      PID USER      PR  NI      VIRT      RES      SHR S %CPU %MEM     TIME+ COMMAND
 2492 student   20   0 1359500 168420  37492 S  0.5  8.2  4:01.04 gnome-shell
 4013 root      20   0  55360  51208   152 S  0.0  2.5  0:00.43 elephant
 1938 root      20   0 189648  35972    7568 R  0.2  1.8  0:30.49 Xorg
 2576 student   20   0  533752  33684  27784 S  0.1  1.6  0:09.29 vmtoolsd
 2420 student   20   0  916268  25616  14404 S  0.0  1.3  0:00.61 gnome-settings-daemon
 2550 student   20   0 1048204  23136  16060 S  0.0  1.1  0:00.46 nautilus
output truncated
```

4. What are the processes with the largest memory allocations?

In addition to the default GNOME shell and **Xorg**, find a process named **elephant**.

5. Turn off the use of bold in the display. Save this configuration for reuse when **top** is restarted.

Press the single uppercase keystroke **B** to toggle bold use off.

Press the single uppercase keystroke **W** to save this configuration. The default configuration file is **.toprc** in the current user's home directory.

6. Exit **top**, then restart it again. Confirm that the new display uses the saved configuration; i.e., the display starts sorted by memory utilization and bold is turned off.

Press **q** to quit the current display, then run **top** again.

```
[student@serverX ~]$ top
```

7. Modify the display to again sort by CPU utilization. Turn on the use of bold. Observe that only *Running* or *Runnable* (state **R**) process entries are bold. Save this configuration.

Press the single uppercase keystroke **P** to sort by CPU utilization.

Press the single uppercase keystroke **B** to toggle bold use on.

Press the single uppercase keystroke **W** to save this configuration.

8. Open another terminal window if necessary. As **root**, suspend the **hippo** process. In **top**, observe that the process state is now **T**.

```
[student@serverX ~]$ su -  
Password: redhat  
[root@serverX ~]# pkill -SIGSTOP hippo
```

9. The **hippo** process quickly disappears from the display, since it is no longer actively using CPU resources. List the process information from the command line to confirm the process state.

```
[root@serverX ~]# ps -f $(pgrep hippo)
```

10. Resume execution of the **hippo** processes.

```
[root@serverX ~]# pkill -SIGCONT hippo
```

11. When finished observing the display, terminate the extra processes using the command line. Confirm that the processes no longer display in **top**.

```
[root@serverX ~]# pkill elephant  
[root@serverX ~]# pkill hippo
```

12. Check that the cleanup is successful by running the grading script. If necessary, find and terminate processes listed by the grading script, and repeat grading.

```
[root@serverX ~]# lab processes grade
```

13. Exit the **top** display. Close extra terminal windows.

Press **q** to quit.

Summary

Processes

Define process components and interpret process viewing commands.

Controlling Jobs

Practice extended process management techniques, including starting, suspending, and connecting to multiple concurrent tasks.

Killing Processes

Use signals to stop, start, and reload processes and process configurations.

Monitoring Process Activity

Manage system workload by utilizing load averages and process statistics.



CHAPTER 8

CONTROLLING SERVICES AND DAEMONS

Overview	
Goal	To control and monitor network services and system daemons using <code>systemd</code> .
Objectives	<ul style="list-style-type: none">• List system daemons and network services started by the <code>systemd</code> service and socket units.• Control system daemons and network services using <code>systemctl</code>.
Sections	<ul style="list-style-type: none">• Identifying Automatically Started System Processes (and Practice)• Controlling System Services (and Practice)
Lab	<ul style="list-style-type: none">• Controlling Services and Daemons

Identifying Automatically Started System Processes

Objectives

After completing this section, students should be able to list system daemons and network services started by the **systemd** service and socket units.

Introduction to **systemd**

System startup and server processes are managed by the *systemd System and Service Manager*. This program provides a method for activating system resources, server daemons, and other processes, both at boot time and on a running system.

Daemons are processes that wait or run in the background performing various tasks. Generally, daemons start automatically at boot time and continue to run until shutdown or until they are manually stopped. By convention, the names of many daemon programs end in the letter "d".

To listen for connections, a daemon uses a **socket**. This is the primary communication channel with local or remote clients. Sockets may be created by daemons or may be separated from the daemon and be created by another process, such as **systemd**. The socket is passed to the daemon when a connection is established by the client.

A service often refers to one or more daemons, but starting or stopping a service may instead make a one-time change to the state of the system, which does not involve leaving a daemon process running afterward (called **oneshot**).

A bit of history

For many years, process ID 1 of Linux and UNIX systems has been the **init** process. This process was responsible for activating other services on the system and is the origin of the term "init system." Frequently used daemons were started on systems at boot time with *System V* and *LSB* init scripts. These are shell scripts, and may vary from one distribution to another. Less frequently used daemons were started on demand by another service, such as **initd** or **xinetd**, which listens for client connections. These systems have several limitations, which are addressed with **systemd**.

In Red Hat Enterprise Linux 7, process ID 1 is **systemd**, the new init system. A few of the new features provided by **systemd** include:

- Parallelization capabilities, which increase the boot speed of a system.
- On-demand starting of daemons without requiring a separate service.
- Automatic service dependency management, which can prevent long timeouts, such as by not starting a network service when the network is not available.
- A method of tracking related processes together by using Linux control groups.



Note

With systemd, shell-based service scripts are used only for a few legacy services. Therefore, configuration files with shell variables, such as those found in **/etc/sysconfig**, are being replaced. Those still in use are included as systemd environment files and read as NAME=VALUE pairs. They are no longer sourced as a shell script.

systemctl and **systemd** units

The **systemctl** command is used to manage different types of systemd objects, called *units*. A list of available unit types can be displayed with **systemctl -t help**.



Important

The **systemctl** may abbreviate or "ellipsize" unit names, process tree entries, and unit descriptions unless run with the **-l** option.

Some common unit types are listed below:

- Service units have a **.service** extension and represent system services. This type of unit is used to start frequently accessed daemons, such as a web server.
- Socket units have a **.socket** extension and represent inter-process communication (IPC) sockets. Control of the socket will be passed to a daemon or newly started service when a client connection is made. Socket units are used to delay the start of a service at boot time and to start less frequently used services on demand. These are similar in principle to services which use the **xinetd** superserver to start on demand.
- Path units have a **.path** extension and are used to delay the activation of a service until a specific file system change occurs. This is commonly used for services which use spool directories, such as a printing system.

Service states

The status of a service can be viewed with **systemctl status name.type**. If the unit type is not provided, **systemctl** will show the status of a service unit, if one exists.

```
[root@serverX ~]# systemctl status sshd.service
sshd.service - OpenSSH server daemon
   Loaded: loaded (/usr/lib/systemd/system/sshd.service; enabled)
   Active: active (running) since Thu 2014-02-27 11:51:39 EST; 7h ago
     Main PID: 1073 (sshd)
      CGroup: /system.slice/sshd.service
              └─1073 /usr/sbin/sshd -D

Feb 27 11:51:39 server0.example.com systemd[1]: Started OpenSSH server daemon.
Feb 27 11:51:39 server0.example.com sshd[1073]: Could not load host key: /et...y
Feb 27 11:51:39 server0.example.com sshd[1073]: Server listening on 0.0.0.0 ....
Feb 27 11:51:39 server0.example.com sshd[1073]: Server listening on :: port 22.
Feb 27 11:53:21 server0.example.com sshd[1270]: error: Could not load host k...y
Feb 27 11:53:22 server0.example.com sshd[1270]: Accepted password for root f...2
Hint: Some lines were ellipsized, use -l to show in full.
```

Several keywords indicating the state of the service can be found in the status output:

Keyword:	Description:
loaded	Unit configuration file has been processed.
active (running)	Running with one or more continuing processes.
active (exited)	Successfully completed a one-time configuration.
active (waiting)	Running but waiting for an event.
inactive	Not running.
enabled	Will be started at boot time.
disabled	Will not be started at boot time.
static	Can not be enabled, but may be started by an enabled unit automatically.



Note

The **systemctl status NAME** command replaces the **service NAME status** command used in previous versions of Red Hat Enterprise Linux.

Listing unit files with systemctl

In this example, please follow along with the next steps while your instructor demonstrates obtaining status information of services.



Note

Notice that the **systemctl** command will automatically paginate the output with **less**.

1. Query the state of all units to verify a system startup.

```
[root@serverX ~]# systemctl
```

2. Query the state of only the service units.

```
[root@serverX ~]# systemctl --type=service
```

3. Investigate any units which are in a failed or maintenance state. Optionally, add the **-l** option to show the full output.

```
[root@serverX ~]# systemctl status rngd.service -l
```

4. The **status** argument may also be used to determine if a particular unit is active and show if the unit is enabled to start at boot time. Alternate commands can also easily show the active and enabled states:

```
[root@serverX ~]# systemctl is-active sshd
[root@serverX ~]# systemctl is-enabled sshd
```

5. List the active state of all loaded units. Optionally, limit the type of unit. The **--all** option will add inactive units.

```
[root@serverX ~]# systemctl list-units --type=service  
[root@serverX ~]# systemctl list-units --type=service --all
```

6. View the enabled and disabled settings for all units. Optionally, limit the type of unit.

```
[root@serverX ~]# systemctl list-unit-files --type=service
```

7. View only failed services.

```
[root@serverX ~]# systemctl --failed --type=service
```



References

systemd(1), **systemd.unit(5)**, **systemd.service(5)**, **systemd.socket(5)**, and **systemctl(1)** man pages

Additional information may be available in the chapter on managing services with **systemd** in the *Red Hat Enterprise Linux System Administrator's Guide* for Red Hat Enterprise Linux 7, which can be found at

<http://docs.redhat.com/>

Practice: Identify the Status of `systemd` Units

Guided exercise

In this lab, you will identify installed and running services on the system.

Outcomes:

A list of active and enabled services on the system.

Before you begin...

Reset your serverX system.

- 1. List all service units on the system.

```
[student@serverX ~]$ sudo systemctl list-units --type=service
```

- 2. List all socket units, active and inactive, on the system.

```
[student@serverX ~]$ sudo systemctl list-units --type=socket --all
```

- 3. Explore the status of the **chrony** service. This service is used for network time synchronization (NTP).

- 3.1. Display the status of the **chrony** service. Note the process ID of any active daemons.

```
[student@serverX ~]$ sudo systemctl status chrony
```

- 3.2. Confirm that the listed daemons are running.

```
[student@serverX ~]$ ps -p PID
```

- 4. Explore the status of the **sshd** service. This service is used for secure encrypted communication between systems.

- 4.1. Determine if the **sshd** service is enabled to start at system boot.

```
[student@serverX ~]$ sudo systemctl is-enabled sshd
```

- 4.2. Determine if the **sshd** service is active without displaying all of the status information.

```
[student@serverX ~]$ sudo systemctl is-active sshd
```

- 4.3. Display the status of the **sshd** service.

```
[student@serverX ~]$ sudo systemctl status sshd
```

5. List the enabled or disabled states of all service units.

```
[student@serverX ~]$ sudo systemctl list-unit-files --type=service
```

Controlling System Services

Objectives

After completing this section, students should be able to control system daemons and network services using **systemctl**.

Starting and stopping system daemons on a running system

Changes to a configuration file or other updates to a service may require that the service be restarted. A service that is no longer used may be stopped before removing the software. A service that is not frequently used may be manually started by an administrator only when it is needed.

In this example, please follow along with the next steps while your instructor demonstrates managing services on a running system.

1. View the status of a service.

```
[root@serverX ~]# systemctl status sshd.service
```

2. Verify that the process is running.

```
[root@serverX ~]# ps -up PID
```

3. Stop the service and verify the status.

```
[root@serverX ~]# systemctl stop sshd.service  
[root@serverX ~]# systemctl status sshd.service
```

4. Start the service and view the status. The process ID has changed.

```
[root@serverX ~]# systemctl start sshd.service  
[root@serverX ~]# systemctl status sshd.service
```

5. Stop, then start, the service in a single command.

```
[root@serverX ~]# systemctl restart sshd.service  
[root@serverX ~]# systemctl status sshd.service
```

6. Issue instructions for a service to read and reload its configuration file without a complete stop and start. The process ID will not change.

```
[root@serverX ~]# systemctl reload sshd.service  
[root@serverX ~]# systemctl status sshd.service
```

Unit dependencies

Services may be started as dependencies of other services. If a socket unit is enabled and the service unit with the same name is not, the service will automatically be started when a request is made on the network socket. Services may also be triggered by path units when a file system condition is met. For example, a file placed into the print spool directory will cause the **cups** print service to be started if it is not running.

```
[root@serverX ~]# systemctl stop cups.service
Warning: Stopping cups, but it can still be activated by:
cups.path
cups.socket
```

To completely stop printing services on a system, stop all three units. Disabling the service will disable the dependencies.

The **systemctl list-dependencies UNIT** command can be used to print out a tree of what other units must be started if the specified unit is started. Depending on the exact dependency, the other unit may need to be running before or after the specified unit starts. The **--reverse** option to this command will show what units need to have the specified unit started in order to run.

Masking services

At times, a system may have conflicting services installed. For example, there are multiple methods to manage networks (network and NetworkManager) and firewalls (iptables and firewalld). To prevent an administrator from accidentally starting a service, that service may be *masked*. Masking will create a link in the configuration directories so that if the service is started, nothing will happen.

```
[root@serverX ~]# systemctl mask network
ln -s '/dev/null' '/etc/systemd/system/network.service'
[root@serverX ~]# systemctl unmask network
rm '/etc/systemd/system/network.service'
```



Important

A disabled service will not be started automatically at boot or by other unit files, but can be started manually. A masked service can not be started manually or automatically.

Enabling system daemons to start or stop at boot

Starting a service on a running system does not guarantee that the service will be started when the system reboots. Similarly, stopping a service on a running system will not keep it from starting again when the system reboots. Services are started at boot time when links are created in the appropriate **systemd** configuration directories. These links are created and removed with **systemctl** commands.

In this example, please follow along with the next steps while your instructor demonstrates enabling and disabling services.

1. View the status of a service.

```
[root@serverX ~]# systemctl status sshd.service
```

2. Disable the service and verify the status. Note that disabling a service does not stop the service.

```
[root@serverX ~]# systemctl disable sshd.service  
[root@serverX ~]# systemctl status sshd.service
```

3. Enable the service and verify the status.

```
[root@serverX ~]# systemctl enable sshd.service  
[root@serverX ~]# systemctl is-enabled sshd.service
```

Summary of **systemctl** commands

Services can be started and stopped on a running system and enabled or disabled for automatic start at boot time.

Task:	Command:
View detailed information about a unit state.	systemctl status <i>UNIT</i>
Stop a service on a running system.	systemctl stop <i>UNIT</i>
Start a service on a running system.	systemctl start <i>UNIT</i>
Restart a service on a running system.	systemctl restart <i>UNIT</i>
Reload configuration file of a running service.	systemctl reload <i>UNIT</i>
Completely disable a service from being started, both manually and at boot.	systemctl mask <i>UNIT</i>
Make a masked service available.	systemctl unmask <i>UNIT</i>
Configure a service to start at boot time.	systemctl enable <i>UNIT</i>
Disable a service from starting at boot time.	systemctl disable <i>UNIT</i>
List units which are required and wanted by the specified unit.	systemctl list-dependencies <i>UNIT</i>



References

systemd(1), **systemd.unit(5)**, **systemd.service(5)**, **systemd.socket(5)**, and **systemctl(1)** man pages

Additional information may be available in the chapter on managing services with **systemd** in the *Red Hat Enterprise Linux System Administrator's Guide* for Red Hat Enterprise Linux 7, which can be found at

<http://docs.redhat.com/>

Practice: Using `systemctl` to Manage Services

Guided exercise

In this lab, you will manage a service unit that is already installed on the system.

Outcomes:

The `chronyd` service is disabled and no longer running on the system.

Before you begin...

Reset your serverX system.

- 1. Observe the results of `systemctl restart` and `systemctl reload` commands.
 - 1.1. Display the status of the `sshd` service. Note the process ID of the daemon.

```
[student@serverX ~]$ sudo systemctl status sshd
```

- 1.2. Restart the `sshd` service and view the status. The process ID of the daemon has changed.

```
[student@serverX ~]$ sudo systemctl restart sshd
[student@serverX ~]$ sudo systemctl status sshd
```

- 1.3. Reload the `sshd` service and view the status. The process ID of the daemon has not changed and connections have not been interrupted.

```
[student@serverX ~]$ sudo systemctl reload sshd
[student@serverX ~]$ sudo systemctl status sshd
```

- 2. Verify that the `chronyd` service is running.

```
[student@serverX ~]$ sudo systemctl status chronyd
```

- 3. Stop the `chronyd` service and view the status.

```
[student@serverX ~]$ sudo systemctl stop chronyd
[student@serverX ~]$ sudo systemctl status chronyd
```

- 4. Determine if the `chronyd` service is enabled to start at system boot.

```
[student@serverX ~]$ sudo systemctl is-enabled chronyd
```

- 5. Reboot the system, then view the status of the `chronyd` service.

```
[student@serverX ~]$ sudo systemctl status chronyd
```

- 6. Disable the **chronyd** service so that it does not start at system boot, then view the status of the service.

```
[student@serverX ~]$ sudo systemctl disable chronyd  
[student@serverX ~]$ sudo systemctl status chronyd
```

- 7. Reboot the system, then view the status of the **chronyd** service.

```
[student@serverX ~]$ sudo systemctl status chronyd
```

Lab: Controlling Services and Daemons

Performance checklist

In this lab, you will manage a service unit that is already installed on the system.

Outcomes:

The **psacct** service is enabled and running on the system, and the **rsyslog** service is disabled and no longer running on the system.

Before you begin...

Reset your serverX system.

1. Start the **psacct** service.
2. Configure the **psacct** service so that it starts at system boot.
3. Stop the **rsyslog** service.
4. Configure the **rsyslog** service so that it does not start at system boot.
5. Reboot the system, then run **lab services grade** to verify the configuration.

Solution

In this lab, you will manage a service unit that is already installed on the system.

Outcomes:

The **psacct** service is enabled and running on the system, and the **rsyslog** service is disabled and no longer running on the system.

Before you begin...

Reset your serverX system.

1. Start the **psacct** service.

```
[student@serverX ~]$ sudo systemctl start psacct  
[student@serverX ~]$ sudo systemctl status psacct
```

2. Configure the **psacct** service so that it starts at system boot.

```
[student@serverX ~]$ sudo systemctl enable psacct  
[student@serverX ~]$ sudo systemctl status psacct
```

3. Stop the **rsyslog** service.

```
[student@serverX ~]$ sudo systemctl stop rsyslog  
[student@serverX ~]$ sudo systemctl status rsyslog
```

4. Configure the **rsyslog** service so that it does not start at system boot.

```
[student@serverX ~]$ sudo systemctl disable rsyslog  
[student@serverX ~]$ sudo systemctl status rsyslog
```

5. Reboot the system, then run **lab services grade** to verify the configuration.

```
[student@serverX ~]$ lab services grade
```

Summary

Identifying Automatically Started System Processes

Determine the status of system daemons and network services started by **systemd**.

Controlling System Services

Start, stop, and enable services using **systemctl**.



CHAPTER 9

CONFIGURING AND SECURING OPENSSH SERVICE

Overview	
Goal	To configure secure command-line access on remote systems using OpenSSH.
Objectives	<ul style="list-style-type: none">Log into a remote system using ssh to run commands from a shell prompt.Set up ssh to allow secure password-free logins by using a private authentication key file.Customize sshd configuration to restrict direct logins as root or to disable password-based authentication.
Sections	<ul style="list-style-type: none">Accessing the Remote Command Line with SSH (and Practice)Configuring SSH Key-based Authentication (and Practice)Customizing SSH Service Configuration (and Practice)
Lab	<ul style="list-style-type: none">Configuring and Securing OpenSSH Service

Accessing the Remote Command Line with SSH

Objective

After completing this section, students should be able to log into a remote system using ssh to run commands from a shell prompt.

What is the OpenSSH secure shell (SSH)?

The term OpenSSH refers to the software implementation of the **Secure Shell** software used in the system. The OpenSSH **Secure Shell**, **ssh**, is used to securely run a shell on a remote system. If you have a user account on a remote Linux system providing SSH services, **ssh** is the command normally used to remotely log into that system. The **ssh** command can also be used to run an individual command on a remote system.

Secure Shell examples

Here are some examples of **ssh** command syntax for remote login and remote execution:

- Create a remote interactive shell as the current user, then return to your previous shell when done with the **exit** command.

```
[student@host ~]$ ssh remotehost
student@remotehost's password:
[student@remotehost ~]$ exit
Connection to remotehost closed.
[student@host ~]$
```

- Connect to a remote shell as a different user (**remoteuser**) on a selected host (**remotehost**):

```
[student@host ~]$ ssh remoteuser@remotehost
remoteuser@remotehost's password:
[remoteuser@remotehost ~]$
```

- Execute a single command (**hostname**) on a remote host (**remotehost**) and as a remote user (**remoteuser**) in a way that returns the output to the local display:

```
[student@host ~]$ ssh remoteuser@remotehost hostname
remoteuser@remotehost's password:
remotehost.example.com
[student@host ~]$
```

The **w** command displays a list of users currently logged into the computer. This is especially useful to show which users are logged in using **ssh** from which remote locations, and what they are doing.

```
[student@host ~]$ w -f
USER     TTY      FROM          LOGIN@    IDLE   JCPU   PCPU WHAT
student   tty1     :0           Wed08    2days  1:52m  0.07s pam: gdm-passwo
root     tty6     -           12:33    4:14m 16.27s 15.74s -bash
```

```

student pts/0 :0.0          Wed08  5:11   1.63s  1.60s /usr/bin/gnome-
student pts/1 :0.0          Wed08  43:44  14.48s 13.81s vim hello.c
student pts/3 :0.0          Wed14  0.00s  0.06s  0.06s w
visitor  pts/6 server2.example. 09:22  3:14   0.02s  0.02s -bash

```

In this example, user *student* logged in on virtual console 1 (**tty1**) through the graphical login (:0) at about 08:00 on Wednesday. User *student* currently has three pseudo-terminals open (**pts/0**, **pts/1**, and **pts/3**) started by the graphical environment; these are almost certainly terminal windows. In one window, *student* is editing **hello.c**. User *root* is logged in on virtual console 6, starting at 12:33 today. User *visitor* logged in on pseudo-terminal 6 at 09:22 today from the host server2.example.com (note that the name has been truncated), probably using **ssh**, and has been sitting idle at a shell prompt for three minutes and 14 seconds.

SSH host keys

SSH secures communication through public-key encryption. When an **ssh** client connects to an SSH server, before the client logs in, the server sends it a copy of its *public key*. This is used to set up the secure encryption for the communication channel and to authenticate the server to the client.

The first time a user uses **ssh** to connect to a particular server, the **ssh** command stores the server's public key in the user's **~/.ssh/known_hosts** file. Every time the user connects after that, the client makes sure it gets the same public key from the server by comparing the server's entry in the **~/.ssh/known_hosts** file to the public key the server sent. If the keys do *not* match, the client assumes that the network traffic is being hijacked or that the server has been compromised, and breaks the connection.

This means that if a server's public key is changed (because the key was lost due to hard drive failure, or replaced for some legitimate reason), users will need to update their **~/.ssh/known_hosts** files and remove the old entry in order to log in.

- Host IDs are stored in **~/.ssh/known_hosts** on your local client system:

```
$ cat ~/.ssh/known_hosts
remotehost,192.168.0.101 ssh-rsa AAAAB3NzaC1E...
```

- Host keys are stored in **/etc/ssh/ssh_host_key*** on the SSH server.

```
$ ls /etc/ssh/*key*
ssh_host_dsa_key      ssh_host_key           ssh_host_rsa_key
ssh_host_dsa_key.pub  ssh_host_key.pub       ssh_host_rsa_key.pub
```



Note

An even better approach is to add entries matching a server's **ssh_host_*key.pub** files to user **~/.ssh/known_hosts** or the systemwide **/etc/ssh/ssh_known_hosts** in advance when the public keys change. See **ssh-copy-id(1)** for an advanced way to manage SSH keys.



References

Additional information may be available in the chapter on using the ssh utility in the *Red Hat Enterprise Linux System Administrator's Guide* for Red Hat Enterprise Linux 7, which can be found at

<http://docs.redhat.com/>

ssh(1), **w(1)**, and **hostname(1)** man pages

Practice: Accessing the Remote Command Line

Guided exercise

In this lab, students will log into a remote system as different users and execute commands.

Outcomes:

Students will log into a remote system and execute commands with the OpenSSH secure shell.

- 1. Log in as student on your desktopX machine.
- 2. **ssh** to your serverX machine. Accept the host key if asked. The host key is recorded on our local machine to identify the remote machine. The **ssh** command will fail to execute properly if the remote ssh host appears to have a different key than the recorded host key. The host key records are stored in the **known_hosts** file in the **.ssh** directory in the user's home directory on the local system.

```
[student@desktopX ~]$ ssh student@serverX
The authenticity of host 'serverX (172.25.X.11)' can't be established.
ECDSA key fingerprint is 47:bf:82:cd:fa:68:06:ee:d8:83:03:1a:bb:29:14:a3.
Are you sure you want to continue connecting (yes/no)? yes
student@serverX's password: student
```

- 3. Run the **w** command. The output of the **w** clearly indicates we have logged in as user student from desktopX.

```
[student@serverX ~]$ w -f
11:01:23 up 1 day, 19:10, 1 user, load average: 0,0,0
USER   TTY   FROM      LOGIN@ IDLE   JCPU   PCPU   WHAT
student pts/1 desktopX 11:01  0.00s 0.12s 0.09s w
```

- 4. Execute the **exit** command to terminate the secure shell connection.

```
[student@serverX ~]$ exit
[student@desktopX ~]$
```

- 5. This time, **ssh** to your serverX machine as user **root**.

```
[student@desktopX ~]$ ssh root@serverX
root@serverX's password: redhat
[root@serverX ~]#
```

- 6. Run the **w** command again. This time, the output of the **w** shows the active connection to the root user account from desktopX.

```
[root@serverX ~]# w -f
11:01:23 up 1 day, 19:10, 1 user, load average: 0,0,0
USER   TTY   FROM      LOGIN@ IDLE   JCPU   PCPU
```

```
root     pts/2 desktopX 11:09  0.00s 0.13s 0.08s w
```

- 7. Run the **exit** to terminate the secure shell connection.

```
[root@serverX ~]# exit  
[student@desktopX ~]$
```

- 8. There are different reasons why a remote host might have legitimately changed its host key. One common reason is when the remote machine is replaced because of hardware failure, or reinstalled. Usually, it is advisable to only remove the key entry for the particular host in the **known_hosts**. In this case, there is only one host entry in the **known_hosts**, so it can be removed completely. Remove the **known_hosts** file for the user student.

```
[student@desktopX ~]$ rm ~/.ssh/known_hosts
```

- 9. **ssh** to serverX as **root** again. Accept the key, log in, and then exit the session.

```
[student@desktopX ~]$ ssh root@serverX  
The authenticity of host 'serverX (::1)' can't be established.  
ECDSA key fingerprint is 47:bf:82:cd:fa:68:06:ee:d8:83:03:1a:bb:29:14:a3.  
Are you sure you want to continue connecting (yes/no)? yes  
root@serverX's password: redhat  
[root@serverX ~]# exit  
[student@desktopX ~]$
```

- 10. Use **ssh** non-interactively to run the **hostname** command on serverX as **root**.

```
[student@desktopX ~]$ ssh root@serverX hostname  
root@serverX's password: redhat  
serverX.example.com
```

Configuring SSH Key-based Authentication

Objective

After completing this section, students should be able to set up SSH to allow secure logins without passwords by using a private authentication key file.

SSH key-based authentication

Users can authenticate **ssh** logins without a password by using *public key authentication*. **ssh** allows users to authenticate using a private-public key scheme. This means that two keys are generated, a private key and a public key. The private key file is used as the authentication credential, and like a password, must be kept secret and secure. The public key is copied to systems the user wants to log into, and is used to verify the private key. The public key does not need to be secret. An SSH server that has the public key can issue a challenge that can only be answered by a system holding your private key. As a result, you can authenticate using the presence of your key. This allows you to access systems in a way that doesn't require typing a password every time, but is still secure.

Key generation is done using the **ssh-keygen** command. This generates the private key **~/.ssh/id_rsa** and the public key **~/.ssh/id_rsa.pub**.



Note

During key generation, there is the option to specify a passphrase which must be provided in order to access your private key. In the event the private key is stolen, it is very difficult for someone other than the issuer to use it when protected with a passphrase. This adds enough of a time buffer to make a new key pair and remove all references to the old keys before the private key can be used by an attacker who has cracked it.

It is always wise to passphrase-protect the private key since the key allows access to other machines. However, this means the passphrase must be entered whenever the key is used, making the authentication process no longer password-less. This can be avoided using **ssh-agent**, which can be given your passphrase once at the start of the session (using **ssh-add**), so it can provide the passphrase as needed while you stay logged in.

For additional information on the **ssh-agent** command, consult the Red Hat System Administration Guide, Chapter 8.2.4.2.: Configuring ssh-agent.

Once the SSH keys have been generated, they are stored by default in the **.ssh/** directory of your home directory. Permissions should be 600 on the private key and 644 on the public key.

Before key-based authentication can be used, the public key needs to be copied to the destination system. This can be done with **ssh-copy-id**.

```
[student@desktopX ~]$ ssh-copy-id root@desktopY
```

When the key is copied to another system using **ssh-copy-id**, it copies the **~/.ssh/id_rsa.pub** file by default.

SSH key demonstration

- Use **ssh-keygen** to create a public-private key pair.

```
[student@desktopX ~]$ ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/home/student/.ssh/id_rsa): Enter
Created directory '/home/student/.ssh'.
Enter passphrase (empty for no passphrase): redhat
Enter same passphrase again: redhat
Your identification has been saved in /home/student/.ssh/id_rsa.
Your public key has been saved in /home/student/.ssh/id_rsa.pub.
The key fingerprint is:
a4:49:cf:fb:ac:ab:c8:ce:45:33:f2:ad:69:7b:d2:5a student@desktopX.example.com
The key's randomart image is:
+--[ RSA 2048]----+
| |
| |
| . .
| . * S
| + + .
| o.E
| o oo+oo
| .=**ooo
+-----+
```

- Use **ssh-copy-id** to copy the public key to the correct location on a remote system. For example:

```
[student@desktopX ~]$ ssh-copy-id -i ~/.ssh/id_rsa.pub root@serverX.example.com
```

References

Additional information may be available in the chapter on using key-based authentication in the *Red Hat Enterprise Linux System Administrator's Guide* for Red Hat Enterprise Linux 7, which can be found at
<http://docs.redhat.com/>

ssh-keygen(1), ssh-copy-id(1), ssh-agent(1), ssh-add(1) man pages

Practice: Using SSH Key-based Authentication

Guided exercise

In this lab, you will set up SSH key-based authentication.

Outcomes:

Students will set up SSH user key-based authentication to initiate SSH connections.

- 1. Create an SSH key pair as **student** on desktopX using no passphrase.

```
[student@desktopX ~]$ ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/home/student/.ssh/id_rsa): Enter
Created directory '/home/student/.ssh'.
Enter passphrase (empty for no passphrase): Enter
Enter same passphrase again: Enter
Your identification has been saved in /home/student/.ssh/id_rsa.
Your public key has been saved in /home/student/.ssh/id_rsa.pub.
...
```

- 2. Send the SSH public key to the **student** account on serverX.

```
[student@desktopX ~]$ ssh-copy-id serverX
The authenticity of host 'serverX (172.25.X.11)' can't be established.
ECDSA key fingerprint is 33:fa:a1:3c:98:30:ff:f6:d4:99:00:4e:7f:84:3e:c3.
Are you sure you want to continue connecting (yes/no)? yes
/usr/bin/ssh-copy-id: INFO: attempting to log in with the new key(s), to filter
out any that are already installed
/usr/bin/ssh-copy-id: INFO: 1 key(s) remain to be installed -- if you are
prompted now it is to install the new keys
student@serverX's password: student

Number of key(s) added: 1

Now try logging into the machine, with: "ssh 'student@serverX'"
and check to make sure that only the key(s) you wanted were added.
```

- 3. Run the **hostname** command by using **ssh** to display the hostname of the serverX.example.com machine without the need to enter a password.

```
[student@desktopX ~]$ ssh serverX 'hostname'
serverX.example.com
```

Customizing SSH Service Configuration

Objective

After completing this section, students should be able to customize sshd configuration to restrict direct logins as root or to disable password-based authentication.

The OpenSSH server configuration file

While OpenSSH server configuration usually does not require modification, additional security measures are available.

Various aspects of the OpenSSH server can be modified in the configuration file **/etc/ssh/sshd_config**.

Prohibit the root user from logging in using SSH

From a security standpoint, it is advisable to prohibit the root user from directly logging into the system with **ssh**.

- The username **root** exists on every Linux system by default, so a potential attacker only has to guess the password, instead of a valid username and password combination.
- The root user has unrestricted privileges.

The OpenSSH server has an internal configuration file setting to prohibit a system login as user **root**, which is commented out by default in the **/etc/ssh/sshd_config** file:

```
#PermitRootLogin yes
```

By enabling the previous option in the **/etc/ssh/sshd_config** configuration file as follows, the root user will be unable to log into the system using the **ssh** command after the sshd service has been restarted:

```
PermitRootLogin no
```

The sshd service has to be restarted to put the changes into effect:

```
[root@serverX ~]# systemctl restart sshd
```

Another option is to only allow key-based ssh login as root with:

```
PermitRootLogin without-password
```

Prohibit password authentication using SSH

Only allowing key-based logins to the remote command line has various advantages:

- SSH keys are longer than an average password, which adds security.
- Less effort to initiate remote shell access after the initial setup.

There is an option in the **/etc/ssh/sshd_config** configuration file which turns on password authentication by default:

```
>PasswordAuthentication yes
```

To prevent password authentication, the **PasswordAuthentication** option has to be set to **no** and the sshd service needs to be restarted:

```
PasswordAuthentication no
```

Keep in mind that whenever you change the **/etc/ssh/sshd_config** file, the sshd service has to be restarted:

```
[root@serverX ~]# systemctl restart sshd
```



References

ssh(1), sshd_config(5) man pages

Practice: Restricting SSH Logins

Guided exercise

In this lab, you will enable additional security features in OpenSSH.

Outcomes:

Prohibit direct SSH login as root on serverX; prohibit users from using passwords to login through SSH to serverX; public key authentication should still be allowed for regular users.

Before you begin...

Reset the desktopX and serverX systems.

Run **lab ssh setup** on both desktopX and serverX. This will create a user account called **visitor** with a password of **password**.

```
[student@desktopX ~]$ lab ssh setup
```

```
[student@serverX ~]$ lab ssh setup
```

- 1. Generate SSH keys on desktopX, copy the public key to the **student** account on serverX, and verify that the keys are working.
 - 1.1. Generate the SSH keys on desktopX.

```
[student@desktopX ~]$ ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/home/student/.ssh/id_rsa): Enter
Created directory '/home/student/.ssh'.
Enter passphrase (empty for no passphrase): Enter
Enter same passphrase again: Enter
Your identification has been saved in /home/student/.ssh/id_rsa.
Your public key has been saved in /home/student/.ssh/id_rsa.pub.
...
```

- 1.2. Copy the SSH public key to the **student** account on serverX.

```
[student@desktopX ~]$ ssh-copy-id serverX
The authenticity of host 'serverX (172.25.X.11)' can't be established.
ECDSA key fingerprint is 33:fa:a1:3c:98:30:ff:f6:d4:99:00:4e:7f:84:3e:c3.
Are you sure you want to continue connecting (yes/no)? yes

/usr/bin/ssh-copy-id: INFO: attempting to log in with the new key(s), to
filter out any that are already installed
/usr/bin/ssh-copy-id: INFO: 1 key(s) remain to be installed -- if you are
prompted now it is to install the new keys
student@serverX's password: student

Number of key(s) added: 1

Now try logging into the machine, with: "ssh 'student@serverX'"
and check to make sure that only the key(s) you wanted were added.
```

- 1.3. Verify that key-based SSH authentication is working for user student on serverX.

```
[student@desktopX ~]$ ssh student@serverX  
[student@serverX ~]$
```

- 2. Log into the serverX machine and obtain superuser privileges.

```
[student@desktopX ~]$ ssh student@serverX  
[student@serverX ~]$ su -  
Password: redhat  
[root@serverX ~]#
```

- 3. Configure SSH on serverX to prevent root logins.

- 3.1. As user root, edit **/etc/ssh/sshd_config** on serverX so that "PermitRootLogin" is uncommented and set to "no."

```
PermitRootLogin no
```

- 3.2. Restart the SSH service on the serverX machine.

```
[root@serverX ~]# systemctl restart sshd
```

- 3.3. Confirm that **root** cannot log in with SSH, but **student** is permitted to log in.

```
[student@desktopX ~]$ ssh root@serverX  
Password: redhat  
Permission denied, please try again.  
Password: redhat  
Permission denied, please try again.  
Password: redhat  
Permission denied (publickey,gssapi-keyex,gssapi-with-mic,password)  
  
[student@desktopX ~]$ ssh student@serverX  
[student@serverX ~]$
```

- 4. Configure SSH on serverX to prevent password authentication.

- 4.1. Edit the configuration file **/etc/ssh/sshd_config** as user root so that the "PasswordAuthentication" entry is set to "no":

```
PasswordAuthentication no
```

- 4.2. Restart the SSH service.

```
[root@serverX ~]# systemctl restart sshd
```

- 4.3. Confirm that **visitor** cannot log in using a password, but **student** is permitted to log in using the SSH keys created earlier.

```
[student@desktopX ~]$ ssh visitor@serverX  
Permission denied (publickey,gssapi-keyex,gssapi-with-mic).
```

Chapter 9. Configuring and Securing OpenSSH Service

```
[student@desktopX ~]$ ssh student@serverX  
[student@serverX ~]$
```

Lab: Configuring and Securing OpenSSH Service

Performance checklist

In this lab, you will add security measures to the ssh service.

Outcomes:

Students will set up SSH keys, configure and exclusively allow user key-based authentication, and lock down the OpenSSH service to prevent the root user from logging into the system by using SSH.

Before you begin...

Reset the desktopX and serverX systems.

Run **lab ssh setup** as the **student** user on both desktopX and serverX. This will create a user account called **visitor** with a password of **password**.

```
[student@desktopX ~]$ lab ssh setup
```

```
[student@serverX ~]$ lab ssh setup
```

Unless specified, all steps are to be performed as user **visitor**.

1. Generate SSH keys on desktopX for user visitor and copy the public key to the **visitor** account on serverX.
2. Disable ssh login for the root user and password-based SSH authentication on serverX.
3. Verify that user root is not allowed to login to serverX by using ssh, while user visitor is with the private key.

Solution

In this lab, you will add security measures to the ssh service.

Outcomes:

Students will set up SSH keys, configure and exclusively allow user key-based authentication, and lock down the OpenSSH service to prevent the root user from logging into the system by using SSH.

Before you begin...

Reset the desktopX and serverX systems.

Run **lab ssh setup** as the **student** user on both desktopX and serverX. This will create a user account called **visitor** with a password of **password**.

```
[student@desktopX ~]$ lab ssh setup
```

```
[student@serverX ~]$ lab ssh setup
```

Unless specified, all steps are to be performed as user **visitor**.

1. Generate SSH keys on desktopX for user visitor and copy the public key to the **visitor** account on serverX.

- 1.1. Generate a SSH public key on desktopX as user visitor.

```
[visitor@desktopX ~]$ ssh-keygen
```

- 1.2. Install the SSH public key generated previously on desktopX to the **visitor** account on serverX.

```
[visitor@desktopX ~]$ ssh-copy-id serverX
The authenticity of host 'serverX (172.25.X.11)' can't be established.
ECDSA key fingerprint is xx:xx:xx:xx:xx:xx:xx:xx:xx:xx:xx:xx:xx:xx:xx:xx.
Are you sure you want to continue connecting (yes/no)? yes
/usr/bin/ssh-copy-id: INFO: attempting to log in with the new key(s), to filter
out any that are already installed
/usr/bin/ssh-copy-id: INFO: 1 key(s) remain to be installed -- if you are
prompted now it is to install the new keys
visitor@serverX's password: password

Number of key(s) added: 1

Now try logging into the machine, with:    "ssh 'visitor@serverX'"
and check to make sure that only the key(s) you wanted were added.
```

2. Disable ssh login for the root user and password-based SSH authentication on serverX.

- 2.1. Log into the serverX virtual machine as user root.

```
[visitor@desktopX ~]$ ssh root@serverX
```

- 2.2. Customize the ssh service on serverX by disabling SSH connections for the user root and only allow key-based login.

Set the necessary config file parameters in **/etc/ssh/sshd_config**:

```
PermitRootLogin no  
PasswordAuthentication no
```

- 2.3. Restart the sshd service on serverX.

```
[root@serverX ~]# systemctl restart sshd
```

3. Verify that user root is not allowed to login to serverX by using ssh, while user visitor is with the private key.

- 3.1. On a different terminal window on desktopX, validate that user root cannot connect to serverX with the **ssh** command. It should fail because we disabled root logins with the ssh service.

```
[visitor@desktopX ~]$ ssh root@serverX  
Permission denied (publickey,gssapi-keyex,gssapi-with-mic).
```

- 3.2. Try logging in as user student to serverX from desktopX by using ssh. It should fail because we did not add the public key from that user to the student account on the serverX machine.

```
[visitor@desktopX ~]$ ssh student@serverX  
Permission denied (publickey,gssapi-keyex,gssapi-with-mic).
```

- 3.3. Verify the ssh service is still accepting key-based authentication by successfully connecting to serverX as user visitor with the **ssh** command.

```
[visitor@desktopX ~]$ ssh visitor@serverX  
[visitor@serverX ~]$
```

Summary

Accessing the Remote Command Line with SSH

The OpenSSH service is the standard software to securely access the remote command line.

Configuring SSH Key-based Authentication

Utilizing key-based SSH authentication adds additional security to remote systems administration.

Customizing SSH Service Configuration

The configuration of the OpenSSH service, sshd, can be changed by editing the file /etc/ssh/sshd_config and restarting the service with systemctl.



CHAPTER 10

ANALYZING AND STORING LOGS

Overview:	
Goal	To locate and accurately interpret relevant system log files for troubleshooting purposes.
Objectives	<ul style="list-style-type: none">Describe the basic syslog architecture in Red Hat Enterprise Linux 7.Interpret entries in relevant syslog files to troubleshoot problems or review system status.Find and interpret log entries in the systemd journal to troubleshoot problems or review system status.Configure systemd-journald to store its journal on disk rather than in memory.Maintain accurate time synchronization and time zone configuration to ensure correct timestamps in system logs.
Sections	<ul style="list-style-type: none">System Log Architecture (and Practice)Reviewing Syslog Files (and Practice)Reviewing systemd Journal Entries (and Practice)Preserving the systemd Journal (and Practice)Maintaining Accurate Time (and Practice)
Lab	<ul style="list-style-type: none">Analyzing and Storing Logs

System Log Architecture

Objectives

After completing this section, students should be able to describe the basic syslog architecture in Red Hat Enterprise Linux 7.

System logging

Processes and the operating system kernel need to be able to record a log of events that happen. These logs can be useful for auditing the system and troubleshooting problems. By convention, the **/var/log** directory is where these logs are persistently stored.

A standard logging system based on the Syslog protocol is built into Red Hat Enterprise Linux. Many programs use this system to record events and organize them into log files. In Red Hat Enterprise Linux 7, syslog messages are handled by two services, **systemd-journald** and **rsyslog**.

The **systemd-journald** daemon provides an improved log management service that collects messages from the kernel, the early stages of the boot process, standard output and error of daemons as they start up and run, and syslog. It writes these messages to a structured journal of events that, by default, does not persist between reboots. This allows syslog messages and events which are missed by syslog to be collected in one central database. The syslog messages are also forwarded by **systemd-journald** to **rsyslog** for further processing.

The **rsyslog** service then sorts the syslog messages by type (or facility) and priority, and writes them to persistent files in the **/var/log** directory.

The **/var/log** directory holds various system- and service-specific log files maintained by **rsyslog**:

Overview of system log files

Log file	Purpose
/var/log/messages	Most syslog messages are logged here. The exceptions are messages related to authentication and email processing, that periodically run jobs, and those which are purely debugging-related.
/var/log/secure	The log file for security and authentication-related messages and errors.
/var/log/maillog	The log file with mail server-related messages.
/var/log/cron	The log file related to periodically executed tasks.
/var/log/boot.log	Messages related to system startup are logged here.



References

systemd-journald.service(8), **rsyslogd(8)**, and **rsyslog.conf(5)** man pages

Additional information may be available in the *Red Hat Enterprise Linux System Administrator's Guide* for Red Hat Enterprise Linux 7, which can be found at
<http://docs.redhat.com/>

Practice: System Logging Components

Quiz

Match the following items to their counterparts in the table.

/var/log	/var/log/boot.log	/var/log/cron	/var/log/maillog
/var/log/messages	/var/log/secure		

Purpose	Log file
Most syslog messages are logged here. The exceptions are messages related to authentication, email processing, and those which periodically run jobs, or those which are purely debugging-related.	
The log file for security and authentication-related messages and errors.	
The directory to which rsyslog is writing all the log files.	
The log file with mail server-related messages.	
The log file related to periodically executed tasks.	
Messages related to system startup are logged here.	

Solution

Match the following items to their counterparts in the table.

Purpose	Log file
Most syslog messages are logged here. The exceptions are messages related to authentication, email processing, and those which are periodically run jobs, or those which are purely debugging-related.	/var/log/messages
The log file for security and authentication-related messages and errors.	/var/log/secure
The directory to which rsyslog is writing all the log files.	/var/log
The log file with mail server-related messages.	/var/log/maillog
The log file related to periodically executed tasks.	/var/log/cron
Messages related to system startup are logged here.	/var/log/boot.log

Reviewing Syslog Files

Objectives

After completing this section, students should be able to interpret entries in relevant syslog files to troubleshoot problems or review system status.

Syslog files

Many programs use the *syslog* protocol to log events to the system. Each log message is categorized by a facility (the type of message) and a priority (the severity of the message). The facilities which are available are documented by the **rsyslog.conf(5)** man page.

The eight priorities are also standardized and ranked as follows:

Overview of syslog priorities

Code	Priority	Severity
0	emerg	System is unusable.
1	alert	Action must be taken immediately.
2	crit	Critical condition.
3	err	Non-critical error condition.
4	warning	Warning condition.
5	notice	Normal but significant event.
6	info	Informational event.
7	debug	Debugging-level message.

The rsyslogd service uses the facility and priority of log messages to determine how to handle them. This is configured by the file **/etc/rsyslog.conf** and by ***.conf** files in **/etc/rsyslog.d**. Programs and administrators can change **rsyslogd** configuration in a way that will not be overwritten by updates to **rsyslog** by putting customized files with a .conf suffix in the **/etc/rsyslog.d** directory.

The ##### RULES ##### section of **/etc/rsyslog.conf** contains directives that define where log messages are saved. The left side of each line indicates the facility and severity of the log message the directive matches. The rsyslog.conf file can contain the character * as a wild card in the facility and severity field, where it either stands for all facilities or all severities. The right side of each line indicates what file to save the log message in. Log messages are normally saved in files in the **/var/log** directory.



Note

Log files are maintained by the **rsyslog** service, and the **/var/log** directory contains a variety of log files specific to certain services. For example, the Apache Web Server or Samba write their own log files into a corresponding subdirectory of the **/var/log** directory.

A message handled by **rsyslog** can appear in multiple different log files. To prevent that, the severity field can be set to **none**, which means that none of the messages directed to this facility are added to the specified log file.

Instead of logging syslog messages to a file, they can be printed to the terminals of all logged-in users. In the default **rsyslog.conf** file, this is done for all messages that have "emerg" priority.

Sample rules section of **rsyslog.conf**

```
#### 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                                         :omusrmsg:*

# 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
```



Note

The **rsyslog.conf** file is documented by the **rsyslog.conf(5)** man page and by extensive HTML documentation in **/usr/share/doc/rsyslog-*/manual.html** contained in the **rsyslog-doc**, which is available from the Red Hat Enterprise Linux 7 software channel, but not included on the installation medium.

Log file rotation

Logs are "rotated" by the **logrotate** utility to keep them from filling up the file system containing **/var/log/**. When a log file is rotated, it is renamed with an extension indicating the date on which it was rotated: the old **/var/log/messages** file may become **/var/log/messages-20141030** if it is rotated on October 30, 2014. Once the old log file is rotated, a new log file is created and the service that writes to it is notified.

After a certain number of rotations, typically after four weeks, the old log file is discarded to free disk space. A cron job runs the logrotate program daily to see if any logs need to be rotated. Most

log files are rotated weekly, but logrotate rotates some faster, or slower, or when they reach a certain size.

Configuration of logrotate is not covered in this course. For more information, see the **logrotate(8)** man page.

Analyze a syslog entry

The system logs written by **rsyslog** start with the oldest message on top and the newest message at the end of the log file. All log entries in log files managed by **rsyslog** are recorded in a standard format. The following example will explain the anatomy of a log file message in the **/var/log/secure** log file:

```
①Feb 11 20:11:48 ②localhost ③sshd[1433]: ④Failed password for student from  
172.25.0.10 port 59344 ssh2
```

- ① The time stamp when the log entry was recorded.
- ② The host from which the log message was sent.
- ③ The program or process that sent the log message.
- ④ The actual message sent.

Monitor a log file with tail

It is especially helpful for reproducing problems and issues to monitor one or more log files for events. The **tail -f /path/to/file** command outputs the last 10 lines of the file specified and continues to output new lines as they get written to the monitored file.

To monitor for failed login attempts on one terminal, run **ssh** as user root while a user tries to log in to the serverX machine:

```
[root@serverX ~]$ tail -f /var/log/secure  
...  
Feb 10 09:01:13 localhost sshd[2712]: Accepted password for root from 172.25.254.254  
port 56801 ssh2  
Feb 10 09:01:13 localhost sshd[2712]: pam_unix(sshd:session): session opened for user  
root by (uid=0)
```

Send a syslog message with logger

The **logger** command can send messages to the **rsyslog** service. By default, it sends the message to the facility user with severity notice (**user.notice**) unless specified otherwise with the **-p** option. It is especially useful to test changes to the **rsyslog** configuration.

To send a message to **rsyslogd** that gets recorded in the **/var/log/boot.log** log file, execute:

```
[root;@serverX ~]$ logger -p local7.notice "Log entry created on serverX"
```



References

logger(1), **tail(1)**, **rsyslog.conf(5)**, and **logrotate(8)** man pages

rsyslog Manual

- **/usr/share/doc/rsyslog-*/*manual.html** provided by the *rsyslog-doc* package

Additional information may be available in the *Red Hat Enterprise Linux System Administrator's Guide* for Red Hat Enterprise Linux 7, which can be found at
<http://docs.redhat.com/>

Practice: Finding Log Entries

Guided exercise

In this lab, you will reconfigure **rsyslog** to write specific messages to a new log file.

Outcomes:

The **rsyslog** service writes all messages with priority debug to the **/var/log/messages-debug** log file for temporary troubleshooting purposes.

- 1. Configure **rsyslog** on serverX to log all messages with severity debug in the newly created log file **/var/log/messages-debug** by adding the **rsyslog** configuration file **/etc/rsyslog.d/debug.conf**. Verify that a generated debug log message with the **logger** command arrives in the **/var/log/messages-debug** log file.
 - 1.1. Change the **rsyslog** configuration to log all messages with severity debug to **/var/log/messages-debug** on serverX by adding the **/etc/rsyslog.d/debug.conf** file.

```
[root@serverX ~]# echo "*.*.debug /var/log/messages-debug" >/etc/rsyslog.d/debug.conf
```

- 1.2. Restart the rsyslog service on serverX.

```
[root@serverX ~]# systemctl restart rsyslog
```

- 2. Generate a debug log message with the **logger** command and verify that the message gets logged to the log file **/var/log/messages-debug** with the **tail** command on serverX.

- 2.1. Monitor the **/var/log/messages-debug** with the **tail** command on serverX.

```
[root@serverX ~]# tail -f /var/log/messages-debug
```

- 2.2. On a separate terminal window, use the **logger** command to generate a debug message on serverX.

```
[root@serverX ~]# logger -p user.debug "Debug Message Test"
```

- 2.3. Switch back to the terminal still running the **tail -f /var/log/messages-debug** command and verify the message sent with the **logger** command shows up.

```
[root@serverX ~]# tail -f /var/log/messages-debug
...
Feb 13 10:37:44 localhost root: Debug Message Test
```

Reviewing systemd Journal Entries

Objectives

After completing this section, students should be able to find and interpret log entries in the systemd journal to troubleshoot problems or review system status.

Finding events with `journalctl`

The systemd journal stores logging data in a structured, indexed binary file. This data includes extra information about the log event. For syslog events, this can include the facility and priority of the original message, for example.



Important

In Red Hat Enterprise Linux 7, the systemd journal is stored in `/run/log` by default, and its contents are cleared after a reboot. This setting can be changed by the system administrator and is discussed elsewhere in this course.

The `journalctl` command shows the full system journal, starting with the oldest log entry, when run as root user:

```
[root@serverX ~]# journalctl
Feb 13 10:01:01 server1 run-parts(/etc/cron.hourly)[8678]: starting 0yum-hourly.cron
Feb 13 10:01:01 server1 run-parts(/etc/cron.hourly)[8682]: finished 0yum-hourly.cron
Feb 13 10:10:01 server1 systemd[1]: Starting Session 725 of user root.
Feb 13 10:10:01 server1 systemd[1]: Started Session 725 of user root.
Feb 13 10:10:01 server1 CROND[8687]: (root) CMD (/usr/lib64/sa/sa1 1 1)
```

The `journalctl` command highlights in bold text messages of priority notice or warning, and messages of priority error and higher are highlighted in red.

The key to successfully using the journal for troubleshooting and auditing is to limit the journal searches to only show relevant output. In the following paragraphs, various different strategies to reduce the output of journal queries will be introduced.

By default, `journalctl -n` shows the last 10 log entries. It takes an optional parameter for how many of the last log entries should be displayed. To display the last 5 log entries, run:

```
[root@serverX ~]# journalctl -n 5
```

When troubleshooting problems, it is useful to filter the output of the journal by priority of the journal entries. The `journalctl -p` takes either the name or the number of the known priority levels and shows the given levels and all higher-level entries. The priority levels known to `journalctl` are debug, info, notice, warning, err, crit, alert, and emerg.

To filter the output of the `journalctl` command to only list any log entry of priority err or above, run:

```
[root@serverX ~]# journalctl -p err
```

Similar to the **tail -f** command, **journalctl -f** outputs the last 10 lines of the journal and continues to output new journal entries as they get written to the journal.

```
[root@serverX ~]# journalctl -f
```

When looking for specific events, it is useful to limit the output to a specific time frame. The **journalctl** command has two options to limit the output to a specific time range, the **--since** and **--until** options. Both options take a time parameter in the format **YYYY-MM-DD hh:mm:ss**. If the date is omitted, the command assumes the date is today, and if the time part is omitted, the whole day starting at 00:00:00 is assumed. Both options take **yesterday**, **today**, and **tomorrow** as valid parameters in addition to the date and time field.

Output all journal entries that got recorded today:

```
[root@serverX ~]# journalctl --since today
```

Output the journal entries from 10th February 2014 20:30:00 to 13th February 2014 12:00:00:

```
[root@serverX ~]# journalctl --since "2014-02-10 20:30:00" --until "2014-02-13 12:00:00"
```

In addition to the visible content of the journal, there are fields attached to the log entries that can only be seen when verbose output is turned on. All of the displayed extra fields can be used to filter the output of a journal query. This is useful to reduce the output of complex searches for certain events in the journal.

```
[root@serverX ~]# journalctl -o verbose
Thu 2014-02-13 02:06:00.409345 EST [s=0b47abbff995149c191a8e539e18c3f9c;
i=d28;b=1ea26e84667848af9a4a2904a76ff9a5;m=4d6878ff5a;t=4f244525daa67;
x=880bc65783036719]
    _PRIORITY=6
    _UID=0
    _GID=0
    _BOOT_ID=1ea26e84667848af9a4a2904a76ff9a5
    _MACHINE_ID=4513ad59a3b442ffa4b7ea88343fa55f
    _CAP_EFFECTIVE=0000001ffffffffff
    _TRANSPORT=syslog
    _SYSLOG_FACILITY=10
    _SYSLOG_IDENTIFIER=sshd
    _COMM=sshd
    _EXE=/usr/sbin/sshd
    _SYSTEMD_CGROUP=/system.slice/sshd.service
    _SYSTEMD_UNIT=sshd.service
    _SELINUX_CONTEXT=system_u:system_r:sshd_t:s0-s0:c0.c1023
    _HOSTNAME=serverX
    _CMDLINE=sshd: root [priv]
    _SYSLOG_PID=6833
    _PID=6833
    MESSAGE=Failed password for root from 172.25.X.10 port 59371 ssh2
    _SOURCE_REALTIME_TIMESTAMP=1392275160409345
```

Among the more useful options to search for lines relevant to a particular process or event are:

- **_COMM** The name of the command
- **_EXE** The path to the executable for the process

- `_PID` The PID of the process
- `_UID` The UID of the user running the process
- `_SYSTEMD_UNIT` The systemd unit that started the process

More than one of these can be combined. For example, the following query shows all journal entries related to processes started by the systemd unit file `sshd.service`, which also have PID 1182:

```
[root@serverX ~]# journalctl _SYSTEMD_UNIT=sshd.service _PID=1182
```



Note

For a list of commonly used journal fields, consult the `systemd.journal-fields(7)` man page.



References

journalctl(1) and `systemd.journal-fields(7)` man pages

Additional information may be available in the *Red Hat Enterprise Linux System Administrator's Guide* for Red Hat Enterprise Linux 7, which can be found at
<http://docs.redhat.com/>

Practice: Finding Events With journalctl

Guided exercise

In this lab, you will filter the systemd journal for specific criteria.

Outcomes:

Students will practice displaying the **systemd** journal output matching different criteria.

- 1. Output only **systemd** journal messages that originate from the **systemd** process that always runs with process id 1 on serverX.

```
[root@serverX ~]# journalctl _PID=1
```

- 2. Display all **systemd** journal messages that originate from a system service started with user id 81 on serverX.

```
[root@serverX ~]# journalctl _UID=81
```

- 3. Output the journal messages with priority **warning** and above on serverX.

```
[root@serverX ~]# journalctl -p warning
```

- 4. Create a **journalctl** query to show all log events recorded in the previous 10 minutes on serverX. The command assumes a current time of 9:15:00.

```
[root@serverX ~]# journalctl --since 9:05:00 --until 9:15:00
```

- 5. Display only the events originating from the **sshd** service with the system unit file **sshd.service** recorded since 9:00:00 this morning on serverX.

```
[root@serverX ~]# journalctl --since 9:00:00 _SYSTEMD_UNIT="sshd.service"
```

Preserving the systemd Journal

Objectives

After completing this section, students should be able to configure **systemd-journald** to store its journal on disk rather than in memory.

Store the system journal permanently

By default, the systemd journal is kept in **/run/log/journal**, which means it is cleared when the system reboots. The journal is a new mechanism in Red Hat Enterprise Linux 7, and for most installations, a detailed journal that starts with the last boot is sufficient.

If the directory **/var/log/journal** exists, the journal will log to that directory instead. The advantage of this is the historic data will be available immediately at boot. However, even with a persistent journal, not all data will be kept forever. The journal has a built-in log rotation mechanism that will trigger monthly. In addition, by default, the journal will not be allowed to get larger than 10% of the file system it is on, or leave less than 15% of the file system free. These values can be tuned in **/etc/systemd/journald.conf**, and the current limits on the size of the journal are logged when the **systemd-journald** process starts, as can be seen by the following command, which shows the top two lines of **journalctl** output:

```
[root@serverX ~]# journalctl | head -2
-- Logs begin at Wed 2014-03-05 15:13:37 CST, end at Thu 2014-03-06 21:57:54 CST. --
Mar 05 15:13:37 serverX.example.com systemd-journal[94]: Runtime journal is using 8.0M
(max 277.8M, leaving 416.7M of free 2.7G, current limit 277.8M).
```

The systemd journal can be made persistent by creating the directory **/var/log/journal** as user root:

```
[root@serverX ~]# mkdir /var/log/journal
```

Ensure that the **/var/log/journal** directory is owned by the root user and group **systemd-journal**, and has the permissions 2755.

```
[root@serverX ~]# chown root:systemd-journal /var/log/journal
[root@serverX ~]# chmod 2755 /var/log/journal
```

Either a reboot of the system or sending the special signal **USR1** as user root to the **systemd-journald** process is required.

```
[root@serverX ~]# killall -USR1 systemd-journald
```

Since the systemd journal is now persistent across reboots, **journalctl -b** can reduce the output by only showing the log messages since the last boot of the system.

```
[root@serverX ~]# journalctl -b
```



Note

When debugging a system crash with a persistent journal, it is usually required to limit the journal query to the reboot before the crash happened. The **-b** option can be accompanied by a negative number indicating to how many prior system boots the output should be limited. For example, the **journalctl -b -1** limits the output to the previous boot.



References

mkdir(1), **systemd-journald(1)**, and **killall(1)** man pages

Additional information may be available in the *Red Hat Enterprise Linux System Administrator's Guide* for Red Hat Enterprise Linux 7, which can be found at
<http://docs.redhat.com/>

Practice: Configure a Persistent systemd Journal

Guided exercise

In this lab, students will make the systemd journal persistent.

Outcomes:

The **systemd** journal is written to disk.

- 1. Configure the systemd journal to be persistent across reboots.

- 1.1. Configure the directory **/var/log/journal** on serverX.

```
[root@serverX ~]# mkdir /var/log/journal  
[root@serverX ~]# chown root:systemd-journal /var/log/journal  
[root@serverX ~]# chmod 2755 /var/log/journal
```

- 1.2. Send the **USR1** signal to the **systemd-journald** or reboot serverX.

```
[root@serverX ~]# killall -USR1 systemd-journald
```

- 2. To verify the systemd journal is persistent, look for a new directory with the systemd journal log files that have been written to **/var/log/journal**. (The exact files which appear may vary on your system, but the directory should have similar contents to the following example.)

```
[root@serverX ~]# ls /var/log/journal/4513ad59a3b442ffa4b7ea88343fa55f  
system.journal    user-1000.journal
```

Maintaining Accurate Time

Objectives

After completing this section, students should be able to maintain accurate time synchronization and time zone configuration to ensure correct timestamps in system logs.

Set local clocks and time zone

Correct synchronized system time is very important for log file analysis across multiple systems. The *Network Time Protocol (NTP)* is a standard way for machines to provide and obtain correct time information on the Internet. A machine may get accurate time information from public NTP services on the Internet such as the NTP Pool Project. A high-quality hardware clock to serve accurate time to local clients is another option.

The **timedatectl** command shows an overview of the current time-related system settings, including current time, time zone, and NTP synchronization settings of the system.

```
[student@serverX ~]$ timedatectl
    Local time: Thu 2014-02-13 02:16:15 EST
    Universal time: Thu 2014-02-13 07:16:15 UTC
        RTC time: Thu 2014-02-13 07:16:15
      Timezone: America/New_York (EST, -0500)
     NTP enabled: yes
   NTP synchronized: no
     RTC in local TZ: no
       DST active: no
Last DST change: DST ended at
                  Sun 2013-11-03 01:59:59 EDT
                  Sun 2013-11-03 01:00:00 EST
Next DST change: DST begins (the clock jumps one hour forward) at
                  Sun 2014-03-09 01:59:59 EST
                  Sun 2014-03-09 03:00:00 EDT
```

A database with known time zones is available and can be listed with:

```
[student@serverX ~]$ timedatectl list-timezones
Africa/Abidjan
Africa/Accra
Africa/Addis_Ababa
Africa/Algiers
Africa/Asmara
Africa/Bamako
...
```

Time zone names are based on the public "tz" (or "zoneinfo") time zone database maintained by IANA. Time zones are named based on continent or ocean, then typically but not always the largest city within the time zone region. For example, most of the US Mountain time zone is "America/Denver."

Selecting the correct name can be non-intuitive in cases where localities inside the time zone have different daylight saving time rules. For example, in the USA, much of the state of Arizona (US Mountain time) does not have a daylight saving time adjustment at all and is in the time zone "America/Phoenix."

The command **tzselect** is useful for identifying correct zoneinfo time zone names. It interactively prompts the user with questions about the system's location, and outputs the name of the correct time zone. It does not make any changes to the time zone setting of the system.

The system setting for the current time zone can be adjusted as user root:

```
[root@serverX ~]# timedatectl set-timezone America/Phoenix
[root@serverX ~]# timedatectl
    Local time: Thu 2014-02-13 00:23:54 MST
    Universal time: Thu 2014-02-13 07:23:54 UTC
        RTC time: Thu 2014-02-13 07:23:53
        Timezone: America/Phoenix (MST, -0700)
    NTP enabled: yes
NTP synchronized: no
RTC in local TZ: no
    DST active: n/a
```

To change the current time and date settings with the **timedatectl** command, the **set-time** option is available. The time is specified in the "YYYY-MM-DD hh:mm:ss" format, where either date or time can be omitted. To change the time to 09:00:00, run:

```
[root@serverX ~]$ timedatectl set-time 9:00:00
[root@serverX ~]$ timedatectl
    Local time: Thu 2014-02-13 09:00:27 MST
    Universal time: Thu 2014-02-13 16:00:27 UTC
        RTC time: Thu 2014-02-13 16:00:28
        Timezone: America/Phoenix (MST, -0700)
    NTP enabled: yes
NTP synchronized: no
RTC in local TZ: no
    DST active: n/a
```

The **set-ntp** option enables or disables NTP synchronization for automatic time adjustment. The option requires either a **true** or **false** argument to turn it on or off. To turn on NTP synchronization, run:

```
[student@desktopX ~]$ timedatectl set-ntp true
```

Configuring and monitoring chronyd

The **chronyd** service keeps the usually-inaccurate local hardware clock (RTC) on track by synchronizing it to the configured NTP servers, or if no network connectivity is available, to the calculated RTC clock drift which is recorded in the **driftfile** specified in the **/etc/chrony.conf** configuration file.

By default, **chronyd** uses servers from the NTP Pool Project for the time synchronization and does not need additional configuration. It may be useful to change the NTP servers when the machine in question is on an isolated network.

The quality of an NTP time source is determined by the **stratum** value reported by the time source. The **stratum** determines the number of hops the machine is away from a high-performance reference clock. The reference clock is a **stratum 0** time source. An NTP server directly attached to it is a **stratum 1**, while a machine synchronizing time from the NTP server is a **stratum 2** time source.

There are two categories of time sources that can be configured in the **/etc/chrony.conf** configuration file, **server** and **peer**. The **server** is one stratum above the local NTP server, and the **peer** is at the same stratum level. More than one **server** and more than one **peer** can be specified, one per line.

The first argument of the **server** line is the IP address or DNS name of the NTP server. Following the server IP address or name, a series of options for the server can be listed. It is recommended to use the **iburst** option, because after the service starts, four measurements are taken in a short time period for a more accurate initial clock synchronization.

To reconfigure the **chronyd** server to synchronize with `classroom.example.com` instead of the default servers configured in the **/etc/chrony.conf**, remove the other server entries and replace them with the following configuration file entry:

```
# Use public servers from the pool.ntp.org project.  
server classroom.example.com iburst
```

After pointing **chronyd** to the local time source, `classroom.example.com`, the service needs to be restarted:

```
[root@serverX ~]# systemctl restart chronyd
```

The **chronyc** command acts as a client to the **chronyd** service. After setting up NTP synchronization, it is useful to verify the NTP server was used to synchronize the system clock. This can be achieved with the **chronyc sources** command or, for more verbose output with additional explanations about the output, **chronyc sources -v**:

```
[root@serverX ~]$ chronyc sources -v  
210 Number of sources = 1  
  
--- Source mode '^' = server, '=' = peer, '#' = local clock.  
/ .- Source state '*' = current synced, '+' = combined , '-' = not combined,  
| / '?' = unreachable, 'x' = time may be in error, '~' = time too variable.  
|| | .- xxxx [ yyyy ] +/- zzzz  
|| | / xxxx = adjusted offset,  
|| | | yyyy = measured offset,  
|| | | zzzz = estimated error.  
|| |  
MS Name/IP address          Stratum Poll Reach LastRx Last sample  
=====  
^* classroom.example.com      8       6     17    23   -497ns[-7000ns] +/-  956us
```

The * character in the S (Source state) field indicates that `classroom.example.com` server has been used as a time source and is the NTP server the machine is currently synchronized to.



Note

Red Hat Enterprise Linux 6 and earlier use **ntpd** and **ntpq** to manage the NTP configuration. Further information may be found in the documentation for Red Hat Enterprise Linux 6.



References

timedatectl(1), **tzselect(8)**, **chronyd(8)**, **chrony.conf(5)**, and **chronyc(1)** man pages

Additional information may be available in the *Red Hat Enterprise Linux System Administrator's Guide* for Red Hat Enterprise Linux 7, which can be found at
<http://docs.redhat.com/>

NTP Pool Project
<http://www.pool.ntp.org/>

Time Zone Database
<http://www.iana.org/time-zones>

Practice: Adjusting System Time

Quiz

The steps to set up the correct time zone and adjust system clocks by using **timedatectl** and **chronyd** follow. Indicate the order in which the steps should be taken.

- a. Identify the appropriate time zone with the **timedatectl list-timezones** command.
- b. Verify the clock was synchronized against the newly specified NTP source with the **chronyc sources** command.
- c. Point **chronyd** to a new time source by adjusting **/etc/chrony.conf**.
- d. Turn on NTP synchronization.
- e. Review current settings with **timedatectl**.
- f. Set the correct time zone with **timedatectl set-timezone**.
- g. Restart the **chronyd** service.

Solution

The steps to set up the correct time zone and adjust system clocks by using **timedatectl** and **chronyd** follow. Indicate the order in which the steps should be taken.

- 2 a. Identify the appropriate time zone with the **timedatectl list-timezones** command.
- 7 b. Verify the clock was synchronized against the newly specified NTP source with the **chronyc sources** command.
- 5 c. Point **chronyd** to a new time source by adjusting **/etc/chrony.conf**.
- 4 d. Turn on NTP synchronization.
- 1 e. Review current settings with **timedatectl**.
- 3 f. Set the correct time zone with **timedatectl set-timezone**.
- 6 g. Restart the **chronyd** service.

Lab: Analyzing and Storing Logs

Performance checklist

In this lab, students will change the time zone and log all authentication failure log entries into a separate file.

Outcomes:

The time zone setting on the serverX machine is adjusted; all **systemd** journal entries recorded in a given time frame are displayed; all syslog messages with the authpriv facility and severity alert are logged into a separate log file.

Before you begin...

Reset your serverX system.

1. Your serverX machine has been relocated to Jamaica. Change the time zone on the serverX machine to Jamaica and verify the time zone has been changed properly.
2. Display all **systemd** journal entries recorded in the last 30 minutes on serverX.
3. Configure **rsyslogd** by adding a rule to the newly created configuration file **/etc/rsyslog.d/auth-errors.conf** to log all security and authentication messages with the priority alert and higher to the **/var/log/auth-errors** file as well. Test the newly added log directive with the **logger** command.

Solution

In this lab, students will change the time zone and log all authentication failure log entries into a separate file.

Outcomes:

The time zone setting on the serverX machine is adjusted; all **systemd** journal entries recorded in a given time frame are displayed; all syslog messages with the authpriv facility and severity alert are logged into a separate log file.

Before you begin...

Reset your serverX system.

1. Your serverX machine has been relocated to Jamaica. Change the time zone on the serverX machine to Jamaica and verify the time zone has been changed properly.
 - 1.1. Identify the correct time zone for Jamaica on serverX.

```
[root@serverX ~]# timedatectl list-timezones
Africa/Abidjan
Africa/Accra
Africa/Addis_Ababa
Africa/Algiers
Africa/Asmara
...
America/Jamaica
...
```

- 1.2. Change the time zone to Jamaica on serverX.

```
[root@serverX ~]# timedatectl set-timezone America/Jamaica
```

- 1.3. Verify the time zone has been properly set on serverX.

```
[root@serverX ~]# timedatectl
    Local time: Thu 2014-02-13 11:16:59 EST
    Universal time: Thu 2014-02-13 16:16:59 UTC
        RTC time: Thu 2014-02-13 16:17:00
        Timezone: America/Jamaica (EST, -0500)
      NTP enabled: yes
     NTP synchronized: no
       RTC in local TZ: no
        DST active: n/a
```

2. Display all **systemd** journal entries recorded in the last 30 minutes on serverX.

Assuming the current time is 9:30:00, the following command would be used

```
[root@serverX ~]# journalctl --since 9:00:00 --until 9:30:00
```

3. Configure **rsyslogd** by adding a rule to the newly created configuration file **/etc/rsyslog.d/auth-errors.conf** to log all security and authentication messages with the priority alert and higher to the **/var/log/auth-errors** file as well. Test the newly added log directive with the **logger** command.

- 3.1. Add the directive to log **authpriv.alert** syslog messages to the **/var/log/auth-errors** file in the **/etc/rsyslog.d/auth-errors.conf** configuration file.

```
[root@serverX ~]# echo "authpriv.alert /var/log/auth-errors" >/etc/rsyslog.d/auth-errors.conf
```

- 3.2. Restart the **rsyslog** service on serverX.

```
[root@serverX ~]# systemctl restart rsyslog
```

- 3.3. Monitor the newly created log file **/var/log/auth-errors** on serverX for changes in a different terminal window.

```
[root@serverX ~]# tail -f /var/log/auth-errors
```

- 3.4. Use the **logger** to create a new log entry to the **/var/log/auth-errors** on serverX.

```
[root@serverX ~]# logger -p authpriv.alert "Logging test authpriv.alert"
```

- 3.5. Verify the message sent to syslog with the **logger** command appears in the **/var/log/auth-errors** on serverX in the terminal running **tail -f /var/log/auth-errors**.

```
[root@serverX ~]# tail -f /var/log/auth-errors
Feb 13 11:21:53 server1 root: Logging test authpriv.alert
```

Summary

System Log Architecture

The log architecture consists of **systemd-journald** for collecting and **rsyslog** to sort and write log messages to the log files.

Reviewing Syslog Files

The system log files are maintained by **rsyslog**.

Reviewing systemd Journal Entries

The systemd journal provides advanced capabilities to query for events.

Preserving the systemd Journal

Configuring **systemd-journald** to permanently store the journal on disk.

Maintaining Accurate Time

Time synchronization is important for log file analysis.



CHAPTER 11

MANAGING RED HAT ENTERPRISE LINUX NETWORKING

Overview	
Goal	To configure basic IPv4 networking on Red Hat Enterprise Linux systems.
Objectives	<ul style="list-style-type: none">Explain fundamental concepts of computer networking.Test and review current network configuration with basic utilities.Manage network settings and devices with <code>nmcli</code> and NetworkManager.Modify network settings by editing the configuration files.Configure and test system host name and name resolution.
Sections	<ul style="list-style-type: none">Networking Concepts (and Practice)Validating Network Configuration (and Practice)Configuring Networking with <code>nmcli</code> (and Practice)Editing Network Configuration Files (and Practice)Configuring Host Names and Name Resolution (and Practice)
Lab	<ul style="list-style-type: none">Managing Red Hat Enterprise Linux Networking

Networking Concepts

Objectives

After completing this section, students should be able to explain fundamental concepts of computer networking.

IPv4 networking

TCP/IP standards follow a four-layer network model specified in RFC1122.

- **Application**

Each application has specifications for communication so that clients and servers may communicate across platforms. Common protocols include SSH (remote login), HTTPS (secure web), NFS or CIFS (file sharing), and SMTP (electronic mail delivery).

- **Transport**

Transport protocols are TCP and UDP. TCP is a reliable connection-oriented communication, while UDP is a connectionless *datagram* protocol. Application protocols use TCP or UDP ports. A list of well-known and registered ports can be found in the `/etc/services` file.

When a packet is sent on the network, the combination of the service port and IP address forms a socket. Each packet has a source socket and a destination socket. This information can be used when monitoring and filtering.

- **Internet**

The Internet, or network layer, carries data from the source host to the destination host. Each host has an IP address and a prefix used to determine network addresses. Routers are used to connect networks.

ICMP is a control protocol at this layer. Instead of ports, it has types. The **ping** utility is an example of using ICMP packets to test connectivity. **ping** sends an ICMP ECHO_REQUEST packet. A successful **ping** receives an ICMP ECHO_REPLY acknowledgment. An unsuccessful **ping** may receive ICMP error messages such as "destination unreachable" or may not receive any response.

- **Link**

The link, or media access, layer provides the connection to physical media. The most common types of networks are wired Ethernet (802.3) and wireless WLAN (802.11). Each physical device has a hardware address (MAC) which is used to identify the destination of packets on the local network segment.

IP Address:

$$172.17.5.3 = 10101100.00010001.00000101.00000011$$
Netmask:

$$255.255.0.0 = \underbrace{11111111.11111111}_{\text{Prefix: /16}}.00000000.00000000$$

$$\underbrace{10101100.00010001}_{\text{Network}}.\underbrace{00000101.00000011}_{\text{Host}}$$
IP Address:

$$192.168.5.3 = 11000000.10101000.00000101.00000011$$

Prefix: /24

Netmask:

$$255.255.255.0 = \underbrace{11111111.11111111.11111111}_{\text{Prefix: /24}}.00000000$$

$$\underbrace{11000000.10101000}_{\text{Network}}.\underbrace{00000101.00000011}_{\text{Host}}$$

Figure 11.1: IPv4 addresses and netmasks

IPv4 addresses

An IPv4 address is a 32-bit number, normally expressed in decimal as four *octets* ranging in value from 0 to 255, separated by dots. The address is divided into two parts: the *network part* and the *host part*. All hosts on the same subnet, which can talk to each other directly without a router, have the same network part; the network part identifies the subnet. No two hosts on the same subnet can have the same host part; the host part identifies a particular host on a subnet.

In the modern Internet, the size of an IPv4 subnet is variable. To know which part of an IPv4 address is the network part and which is the host part, an administrator must know the *netmask* which is assigned to the subnet. The netmask indicates how many bits of the IPv4 address belong to the subnet. The more bits that are available for the host part, the more hosts can be on the subnet.

The lowest possible address on a subnet (host part is all zeros in binary) is sometimes called the *network address*. The highest possible address on a subnet (host part is all ones in binary) is used for broadcast messages in IPv4, and is called the *broadcast address*.

Network masks are expressed in two forms. The older syntax for a netmask which uses 24 bits for the network part would read 255.255.255.0. A newer syntax, called CIDR notation, would specify a *network prefix* of /24. Both forms convey the same information; namely, how many leading bits in the IP address contribute to its network address.

The examples which follow illustrate how the IP address, prefix (netmask), network part, and host part are related.

Calculating the network address for 192.168.1.107/24

Host addr	192.168.1.107	11000000.10101000.00000001.01101011
Network prefix	/24 (255.255.255.0)	11111111.11111111.11111111.00000000
Network addr	192.168.1.0	11000000.10101000.00000001.00000000
Broadcast addr	192.168.1.255	11000000.10101000.00000001.11111111

Calculating the network address for 10.1.1.18/8

Host addr	10.1.1.18	00001010.00000001.00000001.00010010
Network prefix	/8 (255.0.0.0)	11111111.00000000.00000000.00000000
Network addr	10.0.0.0	00001010.00000000.00000000.00000000
Broadcast addr	10.255.255.255	00001010.11111111.11111111.11111111

Calculating the network address for 172.16.181.23/19

Host addr	172.168.181.23	10101100.10101000.10110101.00010111
Network prefix	/19 (255.255.224.0)	11111111.11111111.11100000.00000000
Network addr	172.168.160.0	10101100.10101000.10100000.00000000
Broadcast addr	172.168.191.255	10101100.10101000.10111111.11111111

The special address 127.0.0.1 always points to the local system ("localhost"), and the network 127.0.0.0/8 belongs to the local system, so that it can talk to itself using network protocols.

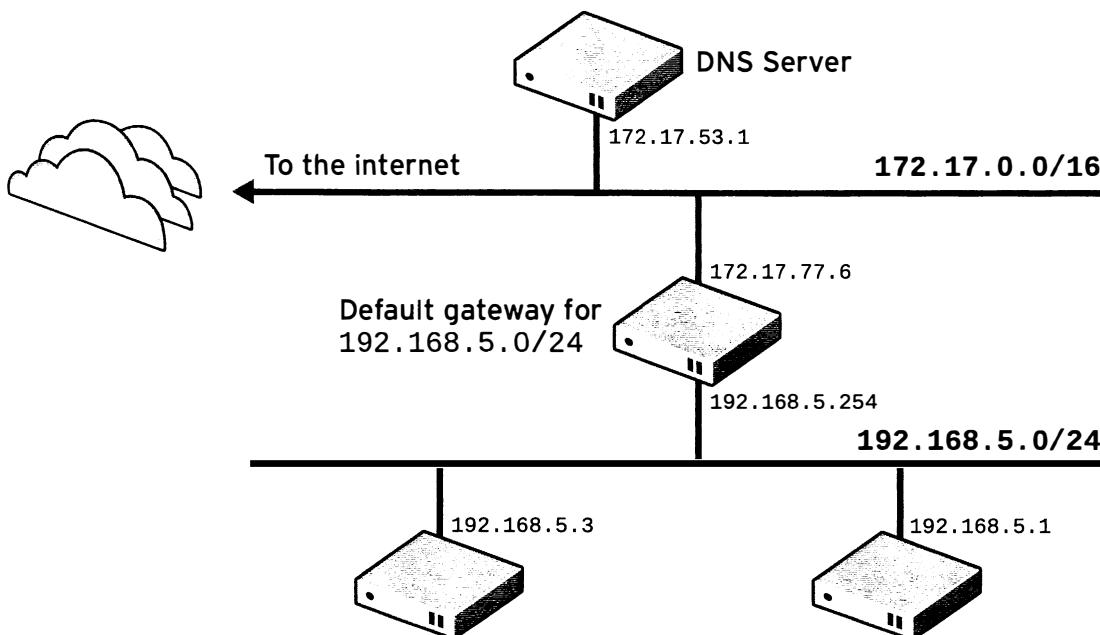


Figure 11.2: Network routing and DNS concepts

IPv4 routing

Whether using IPv4 or IPv6, network traffic needs to move from host to host and network to network. Each host has a *routing table*, which tells it how to route traffic for particular networks.

The routing table entries will list a destination network, which interface to send the traffic out, and the IP address of any intermediate router that is required to relay the message to its final destination. The routing table entry which matches the destination of the network traffic is used to route it. If two entries match, the one with the longest prefix is used.

If the network traffic does not match a more specific route, the routing table usually has an entry for a *default route* to the entire IPv4 Internet, 0.0.0.0/0. This default route points to a *router* on a reachable subnet (that is, on a subnet that has a more specific route in the host's routing table).

If a router receives traffic that is not addressed to it, instead of ignoring it like a normal host, it *forwards* the traffic based on its own routing table. This may send the traffic directly to the destination host (if the router happens to be on the destination's subnet), or it may be forwarded on to another router. This process of forwarding continues until the traffic reaches its final destination.

Example routing table

Destination	Interface	Router (if needed)
192.0.2.0/24	wlo1	
192.168.5.0/24	enp3s0	
0.0.0.0/0 (<i>default</i>)	enp3s0	192.168.5.254

In this example, traffic headed for the IP address 192.0.2.102 from this host will be transmitted directly to that destination via the **wlo1** wireless interface, because it matches the 192.0.2.0/24 route most closely. Traffic for the IP address 192.168.5.3 will be transmitted directly to that destination via the **enp3s0** Ethernet interface, because it matches the 192.168.5.0/24 route most closely.

Traffic to the IP address 10.2.24.1 will be transmitted out the **enp3s0** Ethernet interface to a router at 192.168.5.254, which will forward that traffic on to its final destination. That traffic matches the 0.0.0.0/0 route most closely, as there is not a more specific route in the routing table of this host. The router will use its own routing table to determine where to forward that traffic to next.

Names and IP addresses

The IP protocol uses addresses to communicate, but human beings would rather work with names than long and hard-to-remember strings of numbers. *DNS*, the Domain Name System, is a distributed network of servers that maps host names to IP addresses. In order for name service to work, the host needs to be pointed at a *nameserver*. This nameserver does not need to be on the same subnet; it just needs to be reachable by the host.

DHCP or static network configuration

Many systems are configured to obtain network settings automatically at boot time. The local configuration files indicate that DHCP should be used and a separate client service queries the network for a server and obtains a lease for network settings.

If a DHCP server is not available, the system must use a *static* configuration where the network settings are read from a local configuration file. The correct network settings are obtained from the network administrator or architecture team to ensure there are no conflicts with other systems.

Since DHCP uses the hardware address to track assignments, only one address may be assigned per interface with DHCP. Multiple static addresses may be assigned to a single interface. This

practice is common in systems hosting services for multiple clients, such as HTTP IP-based hosting. Red Hat Enterprise Linux interfaces typically have an IPv4 address and an IPv6 local-link address, but may have more addresses assigned.

Network interface names

Traditionally, network interfaces in Linux are enumerated as **eth0**, **eth1**, **eth2**, and so on. However, the mechanism which sets these names can cause changes to which interface gets which name as devices are added and removed. The default naming behavior in Red Hat Enterprise Linux 7 is to assign fixed names based on firmware, device topology, and device type. Interface names have the following characters:

- Ethernet interfaces begin with *en*, WLAN interfaces begin with *wl*, and WWAN interfaces begin with *ww*.
- The next character(s) represents the type of adapter with an *o* for on-board, *s* for hotplug slot, and *p* for PCI geographic location. Not used by default but also available to administrators, an *x* is used to incorporate a MAC address.
- Finally, a number *N* is used to represent an index, ID, or port.
- If the fixed name cannot be determined, the traditional names such as *ethN* will be used.

For example, the first embedded network interface may be named **eno1** and a PCI card network interface may be named **enp2s0**. The new names make it easier to distinguish the relationship between a port and its name if the user knows both, but the trade off is that users cannot assume a system with one interface calls that interface **eth0**.



Note

Network interface naming can be overridden. If the administrator has installed and enabled the **biosdevname** package or set customized **udev** device naming rules, those settings will override the default naming scheme. Depending on support for **biosdevname** in the system BIOS, names such as **em1**, **em2**, etc. may be used for on-board network cards (corresponding to their names on the chassis). PCI(e) cards are represented with **pYpX** (e.g., **p4p1**), where **Y** is the PCI slot number and **X** is the number for the port on that specific card.



References

services(5), **ping(8)**, **biosdevname(1)**, and **udev(7)** man pages

Additional information may be available in the chapters on configuring networking and consistent network device naming in the *Red Hat Enterprise Linux Networking Guide* for Red Hat Enterprise Linux 7, which can be found at

<http://docs.redhat.com/>

Practice: Networking Concepts

Quiz

Match the following items to their counterparts in the table.

Gateway is not on the same subnet.	
IP address cannot be a network address.	Invalid IPv4 address
Name resolution is not configured.	This configuration is feasible.

Network settings	Correctness
IP address: 172.17.0.351/16 Gateway: 172.17.0.1 DNS server: 172.17.0.254	
IP address: 10.1.2.3/24 Gateway: 10.1.2.1 DNS server: 172.17.4.53	
IP address: 192.168.7.0/24 Gateway: 192.168.7.1 DNS server: 192.168.0.254	
IP address: 10.4.5.6/24 Gateway: 10.4.6.1 DNS server: 192.168.0.254	

Network settings	Correctness
IP address: 172.17.23.5/16 Gateway: 172.17.0.1	

Solution

Match the following items to their counterparts in the table.

Network settings	Correctness
<div style="border: 1px solid black; padding: 5px;"><p>IP address: 172.17.0.351/16 Gateway: 172.17.0.1 DNS server: 172.17.0.254</p></div>	Invalid IPv4 address
<div style="border: 1px solid black; padding: 5px;"><p>IP address: 10.1.2.3/24 Gateway: 10.1.2.1 DNS server: 172.17.4.53</p></div>	This configuration is feasible.
<div style="border: 1px solid black; padding: 5px;"><p>IP address: 192.168.7.0/24 Gateway: 192.168.7.1 DNS server: 192.168.0.254</p></div>	IP address cannot be a network address.
<div style="border: 1px solid black; padding: 5px;"><p>IP address: 10.4.5.6/24 Gateway: 10.4.6.1 DNS server: 192.168.0.254</p></div>	Gateway is not on the same subnet.
<div style="border: 1px solid black; padding: 5px;"><p>IP address: 172.17.23.5/16 Gateway: 172.17.0.1</p></div>	Name resolution is not configured.

Validating Network Configuration

Objectives

After completing this section, students should be able to test and review current network configuration with basic utilities.

Displaying IP addresses

The `/sbin/ip` command is used to show device and address information.

```
[student@desktopX ~]$ ip addr show eth0
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP qlen 1000
    link/ether 52:54:00:00:00:0a brd ff:ff:ff:ff:ff:ff
    ③inet 172.25.0.10/24 brd ④172.25.0.255 scope global eth0
        valid_lft forever preferred_lft forever
    ⑤inet6 fe80::5054:ff:fe00:b/64 scope link
        valid_lft forever preferred_lft forever
```

- ① An active interface has the status of **UP**.
- ② The link line specifies the hardware (MAC) address of the device.
- ③ The inet line shows the IPv4 address and prefix.
- ④ The broadcast address, scope, and device name are also on this line.
- ⑤ The inet6 line shows IPv6 information.

The `ip` command may also be used to show statistics about network performance. The received (RX) and transmitted (TX) packets, errors, and dropped counters can be used to identify network issues caused by congestion, low memory, and overruns.

```
[student@desktopX ~]$ ip -s link show eth0
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP qlen 1000
    link/ether 52:54:00:00:00:0a brd ff:ff:ff:ff:ff:ff
        RX: bytes packets errors dropped overrun mcast
            269850      2931       0       0       0
        TX: bytes packets errors dropped carrier collsns
            300556      3250       0       0       0
```

Troubleshooting routing

The `/sbin/ip` command is also used to show routing information.

```
[student@desktopX ~]$ ip route
default via 172.25.0.254 dev eth0 proto static metric 1024
172.25.X.0/24 dev eth0 proto kernel scope link src 172.25.X.10
10.0.0.0/8 dev eth1 proto kernel scope link src 10.0.0.11
```

All packets destined for the 10.0.0.0/8 network will be sent directly to the destination through the device eth1. All packets destined for the 172.25.X.0/24 network will be sent directly to the destination through the device eth0. All other packets will be sent to the default router located at 172.25.X.254, and also through device eth0.

The **ping** command is used to test connectivity. The command will continue to run until a **Control-C** is pressed unless options are given to limit the number of packets sent.

```
[student@desktopX ~]$ ping -c3 172.25.X.254
```

To trace the path to a remote host, use either **traceroute** or **tracepath**. Both commands can be used to trace a path with UDP packets; however, many networks block UDP and ICMP traffic. The **traceroute** command has options to trace the path with UDP (default), ICMP (-I), or TCP (-T) packets, but may not be installed by default.

```
[student@desktopX ~]$ tracepath access.redhat.com
...
4: 71-32-28-145.rcmt.qwest.net          48.853ms asymm 5
5: dcp-brdr-04.inet.qwest.net           100.732ms asymm 7
6: 206.111.0.153.ptr.us.xo.net         96.245ms asymm 7
7: 207.88.14.162.ptr.us.xo.net         85.270ms asymm 8
8: ae1d0.cir1.atlanta6-ga.us.xo.net    64.160ms asymm 7
9: 216.156.108.98.ptr.us.xo.net        108.652ms
10: bu-ether13.at1ngamq46w-bcr00.tbone.rr.com 107.286ms asymm 12
...
```

Each line in the output of **tracepath** represents a router or *hop* that the packet passes through between the source and the final destination. Additional information is provided as available, including the round trip timing (RTT) and any changes in the maximum transmission unit (MTU) size.

Troubleshooting ports and services

TCP services use sockets as end points for communication and are made up of an IP address, protocol, and port number. Services typically listen on standard ports while clients use a random available port. Well-known names for standard ports are listed in the **/etc/services** file.

The **ss** command is used to display socket statistics. It is similar to the **netstat** command, which is also available but may not be installed by default.

```
[student@desktopX ~]$ ss -ta
State      Recv-Q Send-Q      Local Address:Port          Peer Address:Port
LISTEN      0      128          *:sunrpc                  *:*
LISTEN      0      128          ❶*:ssh                   *:*
LISTEN      0      100          ❷127.0.0.1:smtp          *:*
LISTEN      0      128          *:36889                  *:*
ESTAB       0      0            ❸172.25.X.10:ssh        172.25.254.254:59392
LISTEN      0      128          :::sunrpc                ::::*
LISTEN      0      128          ❹:::ssh                  ::::*
LISTEN      0      100          ❺:::smtp                  ::::*
LISTEN      0      128          :::34946                  ::::*
```

- ❶ The port used for SSH is listening on all IPv4 addresses. The "*" is used to represent "all" when referencing IPv4 addresses or ports.
- ❷ The port used for SMTP is listening on the 127.0.0.1 IPv4 loopback interface.
- ❸ The established SSH connection is on the 172.25.X.10 interface and originates from a system with an address of 172.25.254.254.

- ❸ The port used for SSH is listening on all IPv6 addresses. The ":" syntax is used to represent all IPv6 interfaces.
- ❹ The port used for SMTP is listening on the ::1 IPv6 loopback interface.

Options for ss and netstat

Option	Description
-n	Show numbers instead of names for interfaces and ports.
-t	Show TCP sockets.
-u	Show UDP sockets.
-l	Show only listening sockets.
-a	Show all (listening and established) sockets.
-p	Show the process using the sockets.

References

ip-link(8), ip-address(8), ip-route(8), ip(8), ping(8), tracepath(8), traceroute(8), ss(8), and netstat(8) man pages

Additional information may be available in the chapter on configuring networking in the *Red Hat Enterprise Linux Networking Guide* for Red Hat Enterprise Linux 7, which can be found at

<http://docs.redhat.com/>

Practice: Examining Network Configuration

Guided exercise

In this lab, you will examine the network configuration of the current system.

Outcomes:

Identify the current network interfaces and basic network addresses.

Before you begin...

Reset your serverX system.

- 1. Display the current IP address and netmask for all interfaces.

```
[student@serverX ~]$ ip addr
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN
    link/loopback brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
        valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
        valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP
    link/ether 52:54:00:00:00:0b brd ff:ff:ff:ff:ff:ff
    inet 172.25.X.11/24 brd 172.25.X.255 scope global dynamic eth0
        valid_lft 12704sec preferred_lft 12704sec
    inet6 fe80::5054:ff:fe00:b/64 scope link
        valid_lft forever preferred_lft forever
```

- 2. Display the statistics for the eth0 interface.

```
[student@serverX ~]$ ip -s link show eth0
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP
    mode DEFAULT qlen 1000
    link/ether 52:54:00:00:00:0b brd ff:ff:ff:ff:ff:ff
    RX: bytes packets errors dropped overrun mcast
      418398     4588      0      0      0      0
    TX: bytes packets errors dropped carrier collsns
      360733     1730      0      0      0      0
```

- 3. Display the routing information.

```
[student@serverX ~]$ ip route
default via 172.25.X.254 dev eth0 proto static metric 1024
172.25.X.0/24 dev eth0 proto kernel scope link src 172.25.X.11
```

- 4. Verify that the router is accessible.

```
[student@serverX ~]$ ping -c3 172.25.X.254
PING 172.25.X.254 (172.25.X.254) 56(84) bytes of data.
64 bytes from 172.25.X.254: icmp_seq=1 ttl=64 time=0.489 ms
64 bytes from 172.25.X.254: icmp_seq=2 ttl=64 time=0.510 ms
64 bytes from 172.25.X.254: icmp_seq=3 ttl=64 time=0.458 ms
```

```
--- 172.25.X.254 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 1999ms
rtt min/avg/max/mdev = 0.458/0.485/0.510/0.033 ms
```

5. Show all the hops between the local system and classroom.example.com.

```
[student@serverX ~]$ tracepath classroom.example.com
 1: classroom.example.com                                0.522ms !H
   Resume: pmtu 65535
```

6. Display the listening TCP sockets on the local system.

```
[student@serverX ~]$ ss -lt
State      Recv-Q Send-Q      Local Address:Port          Peer Address:Port
LISTEN     0      128          *:55630                  *:*
LISTEN     0      128          *:sunrpc                 *:*
LISTEN     0      128          *:ssh                   *:*
LISTEN     0      100         127.0.0.1:smtp           *:*
LISTEN     0      128          :::sunrpc                :::*
LISTEN     0      128          :::ssh                  :::*
LISTEN     0      128          :::33079                :::*
LISTEN     0      100          ::1:smtp                :::*
```

Configuring Networking with **nmcli**

Objectives

After completing this section, students should be able to manage network settings and devices with **nmcli** and NetworkManager.

NetworkManager

NetworkManager is a daemon that monitors and manages network settings. In addition to the daemon, there is a GNOME Notification Area applet that provides network status information. Command-line and graphical tools talk to NetworkManager and save configuration files in the **/etc/sysconfig/network-scripts** directory.

A *device* is a network interface. A *connection* is a configuration used for a device which is made up of a collection of settings. Multiple connections may exist for a device, but only one may be active at a time. For example, a system may normally be connected to a network with settings provided by DHCP. Occasionally, that system needs to be connected to a lab or data center network, which only uses static networking. Instead of changing the configuration manually, each configuration can be stored as a separate connection.

Viewing network information with **nmcli**

To display a list of all connections, use **nmcli con show**. To list only the active connections, add the **--active** option.

```
[root@desktopX ~]# nmcli con show
NAME           UUID                                  TYPE      DEVICE
static-eth0    f3e8dd32-3c9d-48f6-9066-551e5b6e612d 802-3-ethernet  eth0
System eth0    5fb06bd0-0bb0-7ffb-45f1-d6edd65f3e03 802-3-ethernet  --
guest         f601ca8a-6647-4188-a431-dab48cc63bf4   802-11-wireless wlp3s0

[root@desktopX ~]# nmcli con show --active
NAME           UUID                                  TYPE      DEVICE
static-eth0    f3e8dd32-3c9d-48f6-9066-551e5b6e612d 802-3-ethernet  eth0
guest         f601ca8a-6647-4188-a431-dab48cc63bf4   802-11-wireless wlp3s0
```

Specify a connection ID (name) to see the details of that connection. The lowercase settings represent the configuration of the connection. Setting and property names are defined in the **nm-settings(5)** man page. The uppercase settings are active data.

```
[root@desktopX ~]# nmcli con show "static-eth0"
...
ipv4.method:          manual
ipv4.dns:             172.25.254.254, 8.8.8.8
ipv4.dns-search:
ipv4.addresses:       { ip = 172.25.X.10/24, gw = 172.25.X.254 }
ipv4.routes:
ipv4.ignore-auto-routes: no
ipv4.ignore-auto-dns:  no
ipv4.dhcp-client-id:  --
ipv4.dhcp-send-hostname: yes
ipv4.dhcp-hostname:   --
ipv4.never-default:  no
ipv4.may-fail:        yes
ipv6.method:          auto
```

...

The **nmcli** command can also be used to show device status and details.

```
[root@desktopX ~]# nmcli dev status
DEVICE  TYPE      STATE      CONNECTION
eth0    ethernet  connected  static-eth0
wlp3s0  wifi      connected  guest
lo     loopback  unmanaged  --
[root@desktopX ~]# nmcli dev show eth0
GENERAL.DEVICE:                eth0
GENERAL.TYPE:                  ethernet
GENERAL.HWADDR:                52:54:00:00:00:0A
GENERAL.MTU:                   1500
GENERAL.STATE:                 100 (connected)
GENERAL.CONNECTION:             static-eth0
GENERAL.CON-PATH:               /org/freedesktop/NetworkManager/
ActiveConnection/1
WIRED-PROPERTIES.CARRIER:       on
IP4.ADDRESS[1]:                ip = 172.25.X.10/24, gw = 172.25.X.254
IP4.DNS[1]:                     172.25.254.254
IP6.ADDRESS[1]:                ip = fe80::5054:ff:fe00:b/64, gw = ::
```

Creating network connections with **nmcli**

When creating a new connection with **nmcli**, the order of the arguments is important. The common arguments appear first and must include the type and interface. Next, specify any type-specific arguments and finally specify the IP address, prefix, and gateway information. Multiple IP addresses may be specified for a single device. Additional settings such as a DNS server are set as modifications once the connection exists.

Examples of creating new connections

Follow along with the next steps while your instructor discusses **nmcli** syntax.

1. Define a new connection named "default" which will autoconnect as an Ethernet connection on the eth0 device using DHCP.

```
[root@desktopX ~]# nmcli con add con-name "default" type ethernet ifname eth0
```

2. Create a new connection named "static" and specify the IP address and gateway. Do not autoconnect.

```
[root@desktopX ~]# nmcli con add con-name "static" ifname eth0 autoconnect no type
                     ethernet ip4 172.25.X.10/24 gw4 172.25.X.254
```

3. The system will autoconnect with the DHCP connection at boot. Change to the static connection.

```
[root@desktopX ~]# nmcli con up "static"
```

4. Change back to the DHCP connection.

```
[root@desktopX ~]# nmcli con up "default"
```



Important

If the static connection is lost, the default connection will attempt to autoconnect. To administratively disable an interface and prevent any autoconnection, use **nmcli dev disconnect DEVICENAME**.

Type options

Type options depend on the type used. An ethernet-type connection may optionally specify a MAC address for the connection. A wifi-type connection must specify the SSID and may specify additional options. Many other types are available, including bridge, bond, team, VPN, and VLAN. To view all the options, use **nmcli con add help**.

```
[root@desktopX ~]# nmcli con add help
Usage: nmcli connection add { ARGUMENTS | help }

ARGUMENTS := COMMON_OPTIONS TYPE_SPECIFIC_OPTIONS IP_OPTIONS

COMMON_OPTIONS:
    type <type>
    ifname <interface name> | "*"
    [con-name <connection name>
     [autoconnect yes|no]

     [save yes|no]

TYPE_SPECIFIC_OPTIONS:
    ethernet:   [mac <MAC address>
                 [cloned-mac <cloned MAC address>
                  [mtu <MTU>
...
...
```

Modifying network interfaces with **nmcli**

An existing connection may be modified with **nmcli con mod** arguments. The arguments are sets of key/value pairs. The key includes a setting name and a property name. Use **nmcli con show "<ID>"** to see a list of current values for a connection. The **nm-settings(5)** man page documents the setting and property names and usage.

```
[root@desktopX ~]# nmcli con show "static"
connection.id:                      static
connection.uuid:                     f3e8dd32-3c9d-48f6-9066-551e5b6e612d
connection.interface-name:           eth0
connection.type:                     802-3-ethernet
connection.autoconnect:              yes
connection.timestamp:                1394905322
connection.read-only:                no
...
```

Examples of connection modifications

Follow along with the next steps while your instructor discusses **nmcli** syntax.

1. Turn off autoconnect.

```
[root@desktopX ~]# nmcli con mod "static" connection.autoconnect no
```

2. Specify a DNS server.

```
[root@desktopX ~]# nmcli con mod "static" ipv4.dns 172.25.X.254
```

3. Some configuration arguments may have values added or removed. Add a +/- symbol in front of the argument. Add a secondary DNS server.

```
[root@desktopX ~]# nmcli con mod "static" +ipv4.dns 8.8.8.8
```

4. Replace the static IP address and gateway.

```
[root@desktopX ~]# nmcli con mod "static" ipv4.addresses "172.25.X.10/24  
172.25.X.254"
```

5. Add a secondary IP address without a gateway.

```
[root@desktopX ~]# nmcli con mod "static" +ipv4.addresses 10.10.10.10/16
```



Important

The **nmcli con mod** will save the setting to the configuration files. To activate the changes, the connection needs to be activated or reactivated.

```
[root@desktopX ~]# nmcli con up "static"
```

Summary of **nmcli** commands

Basic device and connection commands for **nmcli**:

nmcli commands

Command	Use
nmcli dev status	List all devices.
nmcli con show	List all connections.
nmcli con up "<ID>"	Activate a connection.
nmcli con down "<ID>"	Deactivate a connection. The connection will restart if autoconnect is yes.
nmcli dev dis <DEV>	Bring down an interface and temporarily disable autoconnect.
nmcli net off	Disable all managed interfaces.
nmcli con add ...	Add a new connection.
nmcli con mod "<ID>" ...	Modify a connection.
nmcli con del "<ID>"	Delete a connection.



Note

The **nmcli** command also has an interactive edit mode. For a graphical interface, use **nm-connection-editor**.



References

nmcli(1), **nmcli-examples(5)**, and **nm-settings(5)** man pages

Additional information may be available in the section on using the NetworkManager command line tool nmcli in the *Red Hat Enterprise Linux Networking Guide* for Red Hat Enterprise Linux 7, which can be found at

<http://docs.redhat.com/>

Practice: Configuring Networking with **nmcli**

Guided exercise

In this lab, you will configure network settings using **nmcli**.

Outcomes:

Convert a system from DHCP to static configuration.

Before you begin...

Reset your serverX system.

- 1. View network settings using **nmcli**.

- 1.1. Show all connections.

```
[student@serverX ~]$ nmcli con show
NAME           UUID                                  TYPE      DEVICE
System eth0    5fb06bd0-0bb0-7ffb-45f1-d6edd65f3e03  802-3-ethernet  eth0
```

- 1.2. Display all configuration settings for the active connection.

```
[student@serverX ~]$ nmcli con show "System eth0"
connection.id:                         System eth0
connection.uuid:                        5fb06bd0-0bb0-7ffb-45f1-
d6edd65f3e03
connection.interface-name:              eth0
connection.type:                        802-3-ethernet
connection.autoconnect:                 yes
connection.timestamp:                  1394813303
connection.read-only:                  no
connection.permissions:
...
IP4.ADDRESS[1]:                         ip = 172.25.X.11/24, gw =
172.25.X.254
IP4.DNS[1]:                            172.25.254.254
IP4.DOMAIN[1]:                          example.com
...
```

- 1.3. Show device status.

```
[student@serverX ~]$ nmcli dev status
DEVICE  TYPE      STATE       CONNECTION
eth0    ethernet  connected   System eth0
lo     loopback  unmanaged  --
```

- 1.4. Display the settings for the eth0 device.

```
[student@serverX ~]$ nmcli dev show eth0
GENERAL.DEVICE:                      eth0
GENERAL.TYPE:                        ethernet
GENERAL.HWADDR:                      52:54:00:00:00:0B
GENERAL.MTU:                          1500
GENERAL.STATE:                       100 (connected)
```

GENERAL.CONNECTION:	System eth0
GENERAL.CON-PATH:	/org/freedesktop/NetworkManager/
ActiveConnection/1	
WIRED-PROPERTIES.CARRIER:	on
IP4.ADDRESS[1]:	ip = 172.25.X.11/24, gw =
172.25.X.254	
IP4.DNS[1]:	172.25.254.254
IP4.DOMAIN[1]:	example.com
IP6.ADDRESS[1]:	ip = fe80::5054:ff:fe00:b/64, gw =
= ::	

2. Create a static connection with the same IPv4 address, network prefix, and default gateway. Name the new connection *static-eth0*.

```
[student@serverX ~]$ sudo nmcli con add con-name "static-eth0" ifname eth0 type
ethernet ip4 172.25.X.11/24 gw4 172.25.X.254
Connection 'static-eth0' (f3e8dd32-3c9d-48f6-9066-551e5b6e612d) successfully
added.
```

3. Modify the new connection to add the DNS setting.

```
[student@serverX ~]$ sudo nmcli con mod "static-eth0" ipv4.dns 172.25.254.254
```

4. Display and activate the new connection.

- 4.1. View all connections.

NAME	UUID	TYPE	DEVICE
static-eth0	f3e8dd32-3c9d-48f6-9066-551e5b6e612d	802-3-ethernet	--
System eth0	5fb06bd0-0bb0-7ffb-45f1-d6edd65f3e03	802-3-ethernet	eth0

- 4.2. View the active connection.

```
[student@serverX ~]$ nmcli con show --active
System eth0 5fb06bd0-0bb0-7ffb-45f1-d6edd65f3e03 802-3-ethernet eth0
```

- 4.3. Activate the new connection.

```
[student@serverX ~]$ sudo nmcli con up "static-eth0"
Connection successfully activated (D-Bus active path: /org/freedesktop/
NetworkManager/ActiveConnection/3)
```

- 4.4. View the active connection.

NAME	UUID	TYPE	DEVICE
static-eth0	f3e8dd32-3c9d-48f6-9066-551e5b6e612d	802-3-ethernet	eth0

5. Test the connectivity using the new network addresses.

- 5.1. Verify the IP address.

```
[student@serverX ~]$ ip addr show eth0
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP qlen 1000
    link/ether 52:54:00:00:00:0b brd ff:ff:ff:ff:ff:ff
        inet 172.25.X.11/24 brd 172.25.X.255 scope global eth0
            valid_lft forever preferred_lft forever
        inet6 fe80::5054:ff:fe00:b/64 scope link
            valid_lft forever preferred_lft forever
```

- 5.2. Verify the default gateway.

```
[student@serverX ~]$ ip route
default via 172.25.X.254 dev eth0 proto static metric 1024
172.25.X.0/24 dev eth0 proto kernel scope link src 172.25.X.11
```

- 5.3. Ping the DNS address.

```
[student@serverX ~]$ ping -c3 172.25.254.254
PING 172.25.254.254 (172.25.254.254) 56(84) bytes of data.
64 bytes from 172.25.254.254: icmp_seq=1 ttl=64 time=0.419 ms
64 bytes from 172.25.254.254: icmp_seq=2 ttl=64 time=0.598 ms
64 bytes from 172.25.254.254: icmp_seq=3 ttl=64 time=0.503 ms

--- 172.25.254.254 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 1999ms
rtt min/avg/max/mdev = 0.419/0.506/0.598/0.077 ms
```

- 6. Configure the original connection so that it does not start at boot and verify that the static connection is used when the system reboots.

- 6.1. Disable the original connection from autostarting at boot.

```
[student@serverX ~]$ sudo nmcli con mod "System eth0" \
> connection.autoconnect no
```

- 6.2. Reboot the system.

```
[student@serverX ~]$ reboot
```

- 6.3. View the active connection.

```
[student@serverX ~]$ nmcli con show --active
NAME           UUID                                  TYPE      DEVICE
static-eth0    f3e8dd32-3c9d-48f6-9066-551e5b6e612d 802-3-ethernet  eth0
```

Editing Network Configuration Files

Objectives

After completing this section, students should be able to modify network settings by editing the configuration files.

Modifying network configuration

It is also possible to configure the network by editing interface configuration files. Interface configuration files control the software interfaces for individual network devices. These files are usually named **/etc/sysconfig/network-scripts/ifcfg-<name>**, where <name> refers to the name of the device or connection that the configuration file controls. The following are standard variables found in the file used for static or dynamic configuration.

Configuration Options for ifcfg File

<i>Static</i>	<i>Dynamic</i>	<i>Either</i>
BOOTPROTO=none	BOOTPROTO=dhcp	DEVICE=eth0
IPADDR0=172.25.X.10		NAME="System eth0"
PREFIX0=24		ONBOOT=yes
GATEWAY0=172.25.X.254		UUID=f3e8dd32-3...
DEFROUTE=yes		USERCTL=yes
DNS1=172.25.254.254		

In the static settings, variables for IP address, prefix, and gateway have a number at the end. This allows multiple sets of values to be assigned to the interface. The DNS variable also has a number which is used to specify the order of lookup when multiple servers are specified.

After modifying the configuration files, run **nmcli con reload** to make NetworkManager read the configuration changes. The interface still needs to be restarted for changes to take effect.

```
[root@serverX ~]# nmcli con reload
[root@serverX ~]# nmcli con down "System eth0"
[root@serverX ~]# nmcli con up "System eth0"
```



References

nmcli(1) man page

Additional information may be available in the chapter on configuring networking in the *Red Hat Enterprise Linux Networking Guide* for Red Hat Enterprise Linux 7, which can be found at

<http://docs.redhat.com/>

Practice: Editing Network Configuration Files

Guided exercise

In this lab, you will edit network configuration files.

Outcomes:

An additional network address added to each system.

Before you begin...

Reset your serverX and desktopX systems.

- 1. As the root user, edit the **/etc/sysconfig/network-scripts/ifcfg-eth0** on serverX to add an additional address of **10.0.X.1/24**.
 - 1.1. Append an entry to the file to specify the IPv4 address.

```
[root@serverX ~]# echo "IPADDR1=10.0.X.1" >> /etc/sysconfig/network-scripts/ifcfg-eth0
```

- 1.2. Append an entry to the file to specify the network prefix.

```
[root@serverX ~]# echo "PREFIX1=24" >> /etc/sysconfig/network-scripts/ifcfg-eth0
```

- 2. Activate the new address.

- 2.1. Reload the configuration changes.

```
[root@serverX ~]# nmcli con reload
```

- 2.2. Restart the connection with the new settings.

```
[root@serverX ~]# nmcli con up "System eth0"
```

- 3. As the root user, edit the **/etc/sysconfig/network-scripts/ifcfg-eth0** on desktopX to add an additional address of **10.0.X.2/24** and load the new configuration.

- 3.1. Modify the file to add the IPv4 and network prefix.

```
[root@desktopX ~]# echo "IPADDR1=10.0.X.2" >> /etc/sysconfig/network-scripts/ifcfg-eth0
[root@desktopX ~]# echo "PREFIX1=24" >> /etc/sysconfig/network-scripts/ifcfg-eth0
```

- 3.2. Reload the configuration changes.

```
[root@desktopX ~]# nmcli con reload
```

- 3.3. Bring up the connection with the new settings.

```
[root@desktopX ~]# nmcli con up "System eth0"
```

- 4. Test the connectivity using the new network addresses.

- 4.1. On serverX, verify the IP address.

```
[root@serverX ~]# ip addr
```

- 4.2. On serverX, ping the new address of desktopX.

```
[root@serverX ~]# ping 10.0.X.2
```

- 4.3. On desktopX, verify the IP address.

```
[root@desktopX ~]# ip addr
```

- 4.4. On desktopX, ping the new address of serverX.

```
[root@desktopX ~]# ping 10.0.X.1
```

Configuring Host Names and Name Resolution

Objectives

After completing this section, students should be able to configure and test system host name and name resolution.

Changing the system host name

The **hostname** command displays or temporarily modifies the system's fully qualified host name.

```
[root@desktopX ~]# hostname  
desktopX.example.com
```

A static host name may be specified in the **/etc/hostname** file. The **hostnamectl** command is used to modify this file and may be used to view the status of the system's fully qualified host name. If this file does not exist, the host name is set by a reverse DNS query once the interface has an IP address assigned.

```
[root@desktopX ~]# hostnamectl set-hostname desktopX.example.com  
[root@desktopX ~]# hostnamectl status  
  Static hostname: desktopX.example.com  
    Icon name: computer  
    Chassis: n/a  
  Machine ID: 9f6fb63045a845d79e5e870b914c61c9  
    Boot ID: aa6c3259825e4b8c92bd0f601089ddf7  
Virtualization: kvm  
Operating System: Red Hat Enterprise Linux Server 7.0 (Maipo)  
  CPE OS Name: cpe:/o:redhat:enterprise_linux:7.0:beta:server  
    Kernel: Linux 3.10.0-97.el7.x86_64  
  Architecture: x86_64  
[root@desktopX ~]# cat /etc/hostname  
desktopX.example.com
```



Important

The static host name is stored in **/etc/hostname**. Previous versions of Red Hat Enterprise Linux stored the host name as a variable in the **/etc/sysconfig/network** file.

Configuring name resolution

The **stub resolver** is used to convert host names to IP addresses or the reverse. The contents of the file **/etc/hosts** are checked first.

```
[root@desktopX ~]# cat /etc/hosts  
127.0.0.1      localhost localhost.localdomain localhost4 localhost4.localdomain4  
::1            localhost localhost.localdomain localhost6 localhost6.localdomain6  
  
172.25.254.254 classroom.example.com  
172.25.254.254 content.example.com
```

The **getent hosts *hostname*** command can be used to test host name resolution with the **/etc/hosts** file.

If an entry is not found in that file, the stub resolver looks for the information from a DNS nameserver. The **/etc/resolv.conf** file controls how this query is done:

- **nameserver**: the IP address of a nameserver to query. Up to three nameserver directives may be given to provide backups if one is down.
- **search**: a list of domain names to try with a short host name. Both this and **domain** should not be set in the same file; if they are, the last instance wins. See **resolv.conf(5)** for details.

```
[root@desktopX ~]# cat /etc/resolv.conf
# Generated by NetworkManager
domain example.com
search example.com
nameserver 172.25.254.254
```

NetworkManager will update the **/etc/resolv.conf** file using DNS settings in the connection configuration files. Use the **nmcli** to modify the connections.

```
[root@desktopX ~]# nmcli con mod ID ipv4.dns IP
[root@desktopX ~]# nmcli con down ID
[root@desktopX ~]# nmcli con up ID
[root@desktopX ~]# cat /etc/sysconfig/network-scripts/ifcfg-ID
...
DNS1=8.8.8.8
...
```

The default behavior of **nmcli con mod ID ipv4.dns IP** is to replace any previous DNS settings with the new IP list provided. A +/- symbol in front of the **ipv4.dns** argument will add or remove an individual entry.

```
[root@desktopX ~]# nmcli con mod ID +ipv4.dns IP
```

The **host *HOSTNAME*** command can be used to test DNS server connectivity.

```
[root@desktopX ~]# host classroom.example.com
classroom.example.com has address 172.25.254.254
[root@desktopX ~]# host 172.25.254.254
254.254.25.172.in-addr.arpa domain name pointer classroom.example.com.
```



Important

If DHCP is in use, **/etc/resolv.conf** is automatically rewritten as interfaces are started, unless you specify **PEERDNS=no** in the relevant interface configuration files. The change can be made with **nmcli**.

```
[root@desktopX ~]# nmcli con mod "System eth0" ipv4.ignore-auto-dns yes
```



References

nmcli(1), **hostnamectl(1)**, **hosts(5)**, **getent(1)**, **host(1)**, and **resolv.conf(5)** man pages

Additional information may be available in the chapter on configuring host names in the *Red Hat Enterprise Linux Networking Guide* for Red Hat Enterprise Linux 7, which can be found at

<http://docs.redhat.com/>

Practice: Configuring Host Names and Name Resolution

Guided exercise

In this lab, you will configure the system host name and name resolution.

Outcomes:

Customized host name and name resolution settings.

Before you begin...

Reset your serverX system.

- 1. View the current host name settings.

- 1.1. Display the current host name.

```
[student@serverX ~]$ hostname  
serverX.example.com
```

- 1.2. Display the host name status.

```
[student@serverX ~]$ hostnamectl status  
      Static hostname: n/a  
        Transient hostname: serverX.example.com  
          Icon name: computer  
            Chassis: n/a  
      Machine ID: 9f6fb63045a845d79e5e870b914c61c9  
        Boot ID: d4ec3a2e8d3c48749aa82738c0ea946a  
Operating System: Red Hat Enterprise Linux Server 7.0 (Maipo)  
      CPE OS Name: cpe:/o:redhat:enterprise_linux:7.0:beta:server  
        Kernel: Linux 3.10.0-97.el7.x86_64  
      Architecture: x86_64
```

- 2. Set a static host name to match the current transient host name.

- 2.1. Change the host name and host name configuration file. Replace the X with your station number and match the output of the previous step.

```
[student@serverX ~]$ sudo hostnamectl set-hostname serverX.example.com
```

- 2.2. View the configuration file providing the host name at network start.

```
[student@serverX ~]$ cat /etc/hostname  
serverX.example.com
```

- 2.3. Display the host name status.

```
[student@serverX ~]$ hostnamectl status  
      Static hostname: serverX.example.com  
          Icon name: computer
```

Chapter 11. Managing Red Hat Enterprise Linux Networking

```
Chassis: n/a
Machine ID: 9f6fb63045a845d79e5e870b914c61c9
Boot ID: d4ec3a2e8d3c48749aa82738c0ea946a
Operating System: Red Hat Enterprise Linux Server 7.0 (Maipo)
CPE OS Name: cpe:/o:redhat:enterprise_linux:7.0:beta:server
Kernel: Linux 3.10.0-97.el7.x86_64
Architecture: x86_64
```

- 3. Temporarily change the host name.

- 3.1. Change the host name.

```
[student@serverX ~]$ sudo hostname testname
```

- 3.2. Display the current host name.

```
[student@serverX ~]$ hostname
testname
```

- 3.3. View the configuration file providing the host name at network start.

```
[student@serverX ~]$ cat /etc/hostname
serverX.example.com
```

- 3.4. Reboot the system.

```
[student@serverX ~]$ reboot
```

- 3.5. Display the current host name.

```
[student@serverX ~]$ hostname
serverX.example.com
```

- 4. Add a local nickname for the classroom server.

- 4.1. Look up the IP address of the classroom.example.com.

```
[student@serverX ~]$ host classroom.example.com
classroom.example.com has address 172.25.254.254
```

- 4.2. Modify **/etc/hosts** so that the name **class** has the IP address 172.25.254.254 and can be used to communicate with classroom.example.com.

```
[student@serverX ~]$ sudo vi /etc/hosts
[student@serverX ~]$ cat /etc/hosts
127.0.0.1 localhost localhost.localdomain localhost4
localhost4.localdomain4
::1 localhost localhost.localdomain localhost6
localhost6.localdomain6

172.25.254.254 classroom.example.com class
```

```
172.25.254.254 content.example.com
```

- 4.3. Look up the IP address of the class.

```
[student@serverX ~]$ host class
Host class not found: 2(SERVFAIL)
[student@serverX ~]$ getent hosts class
172.25.254.254    classroom.example.com class
```

- 4.4. Ping class.

```
[student@serverX ~]$ ping -c3 class
PING classroom.example.com (172.25.254.254) 56(84) bytes of data.
64 bytes from classroom.example.com (172.25.254.254): icmp_seq=1 ttl=64
time=0.397 ms
64 bytes from classroom.example.com (172.25.254.254): icmp_seq=2 ttl=64
time=0.447 ms
64 bytes from classroom.example.com (172.25.254.254): icmp_seq=3 ttl=64
time=0.470 ms

--- classroom.example.com ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2000ms
rtt min/avg/max/mdev = 0.397/0.438/0.470/0.030 ms
```

Lab: Managing Red Hat Enterprise Linux Networking

Performance checklist

In this lab, you will configure basic IPv4 networking on Red Hat Enterprise Linux systems.

Outcomes:

The primary interface has two static IPv4 addresses configured.

Before you begin...

Reset your desktopX system.

1. Create a new connection with a static network connection using the settings in the table. Be sure to replace the X with the correct number for your systems.

Parameter	Setting
Connection name	lab
IP address	172.25.X.10/24
Gateway address	172.25.X.254
DNS address	172.25.254.254

2. Configure the new connection to be autostarted. Other connections should not start automatically.
3. Modify the new connection so that it also uses the address 10.0.X.1/24.
4. Configure the **hosts** file so that 10.0.X.1 can be referenced as "private".
5. Reboot the system, then run **lab network grade** to verify settings.

Solution

In this lab, you will configure basic IPv4 networking on Red Hat Enterprise Linux systems.

Outcomes:

The primary interface has two static IPv4 addresses configured.

Before you begin...

Reset your desktopX system.

1. Create a new connection with a static network connection using the settings in the table. Be sure to replace the X with the correct number for your systems.

Parameter	Setting
Connection name	lab
IP address	172.25.X.10/24
Gateway address	172.25.X.254
DNS address	172.25.254.254

```
[root@desktopX ~]# nmcli con add con-name lab ifname eth0 type ethernet ip4
172.25.X.10/24 gw4 172.25.X.254
[root@desktopX ~]# nmcli con mod "lab" ipv4.dns 172.25.254.254
```

2. Configure the new connection to be autostarted. Other connections should not start automatically.

```
[root@desktopX ~]# nmcli con mod "lab" connection.autoconnect yes
[root@desktopX ~]# nmcli con mod "System eth0" connection.autoconnect no
```

3. Modify the new connection so that it also uses the address 10.0.X.1/24.

```
[root@desktopX ~]# nmcli con mod "lab" +ipv4.addresses 10.0.X.1/24
```

Or alternately:

```
[root@desktopX ~]# echo "IPADDR1=10.0.X.1" >> /etc/sysconfig/network-scripts/ifcfg-
lab
[root@desktopX ~]# echo "PREFIX1=24" >> /etc/sysconfig/network-scripts/ifcfg-lab
```

4. Configure the **hosts** file so that 10.0.X.1 can be referenced as "private".

```
[root@desktopX ~]# echo "10.0.X.1 private" >> /etc/hosts
```

5. Reboot the system, then run **lab network grade** to verify settings.

```
[root@desktopX ~]# lab network grade
```

Summary

Networking Concepts

List features of computer networking.

Validating Network Configuration

Use basic utilities to determine current network configuration.

Configuring Networking with **nmcli**

Manage network devices with command-line utilities.

Editing Network Configuration Files

Modify network configuration files.

Configuring Host Names and Name Resolution

Display and change system host name and name resolution configuration.



CHAPTER 12

ARCHIVING AND COPYING FILES BETWEEN SYSTEMS

Overview:	
Goal	To archive and copy files from one system to another.
Objectives	<ul style="list-style-type: none">• Use tar to create new compressed archive files and extract files from existing archive files.• Copy files securely to or from a remote system running sshd.• Securely synchronize the contents of a local file or directory with a remote copy.
Sections	<ul style="list-style-type: none">• Managing Compressed tar Archives (and Practice)• Copying Files Between Systems Securely (and Practice)• Synchronizing Files Between Systems Securely (and Practice)
Lab	<ul style="list-style-type: none">• Archiving and Copying Files Between Systems

Managing Compressed tar Archives

Objective

After completing this section, students should be able to use tar to create new compressed archive files and extract files from existing archive files.

What is tar?

Archiving and compressing files are useful when creating backups and transferring data across a network. One of the oldest and most common commands for creating and working with backup archives is the **tar** command.

With **tar**, users can gather large sets of files into a single file (archive). The archive can be compressed using **gzip**, **bzip2**, or **xz** compression.

The **tar** command can list the contents of archives or extract their files to the current system. Examples of how to use the **tar** command are included in this section.

Operate the tar command

To use the **tar** command, one of the three following actions is required:

- **c** (create an archive)
- **t** (list the contents of an archive)
- **x** (extract an archive)

Commonly used options are:

- **f file name** (file name of the archive to operate on)
- **v** (verbosity; useful to see which files get added to or extracted from the archive)



Note

A leading - is not required for tar options.

Archive files and directories with tar

Before creating a tar archive, verify that there is no other archive in the directory with the same name as the new archive to be created. The **tar** command will overwrite an existing archive without any feedback.

The first option to use when creating a new archive is the **c** option, followed by the **f** option, then a single space, then the file name of the archive to be created, and finally the list of files and directories that should get added to the archive. The archive is created in the current directory unless specified otherwise.

In the following example, an archive named **archive.tar** is created with the contents of file1, file2, and file3 in the user's home directory.

```
[user@host ~]# tar cf archive.tar file1 file2 file3
[user@host ~]# ls archive.tar
archive.tar
```



Note

When archiving files by absolute path names, the leading / of the path is removed from the file name by default. This helps avoid mistakes which could cause important files to be overwritten. Files are normally extracted relative to the current working directory of the tar command.

For **tar** to be able to archive the selected files, it is mandatory that the user executing the **tar** command is able to read the file(s). For example, creating a new archive of the **/etc** folder and all of its content requires root privileges, because only root is allowed to read all of the files there. An unprivileged user could create an archive of the **/etc** folder, but the archive would omit files which do not include read permission for the user and it would omit directories which do not include both read and execute permission for the user.

Create the tar archive **/root/etc.tar** with the **/etc** directory as content as user root:

```
[root@host ~]# tar cf /root/etc.tar /etc
tar: Removing leading `/' from member names
[root@host ~]#
```



Important

While **tar** stores ownership and permissions of the files, there are other attributes that are not stored in the **tar** archive by default, such as the SELinux context and ACLs. To store those extended attributes in the **tar** archive, the **--xattrs** option is required when creating an archive.

List contents of a tar archive

To list the content of an archive, the **t** and **f** options, accompanied by the archive to operate, are required.

List the content of the archive **/root/etc.tar**:

```
[root@host ~]# tar tf /root/etc.tar
etc/
etc/fstab
etc/crypttab
etc/mtab
...
```

Extract an archive created with tar

A tar archive should normally be extracted in an empty directory to ensure it does not overwrite any existing files. If files are extracted by root, **tar** attempts to preserve the original user and group ownership of the files. If a regular user extracts files using **tar**, the extracted files are owned by that user.

Extract the archive **/root/etc.tar** to the **/root/etcbackup** directory:

```
[root@host ~]# mkdir /root/etcbackup  
[root@host ~]# cd /root/etcbackup  
[root@host etcbackup]# tar xf /root/etc.tar
```

By default, when files get extracted from an archive, the umask is subtracted from the permissions of archive content. This is a security measure and prevents extracted regular files from receiving execute permissions by default. To preserve the permissions of an archived file, the **p** option is to be used when extracting an archive.

Extract the archive **/root/myscripts.tar** to the **/root/scripts** directory while preserving the permissions of the extracted files:

```
[root@host ~]# mkdir /root/scripts  
[root@host ~]# cd /root/scripts  
[root@host scripts]# tar xpf /root/myscripts.tar
```

Create a compressed tar archive

There are three different compression methods supported by the **tar** command. The **gzip** compression is the fastest and oldest one, and is most widely available. The **bzip2** compression usually leads to smaller archive files compared to **gzip** and is less widely available than **gzip**, while the **xz** compression method is relatively new, but usually offers the best compression ratio of the methods available.



Note

The effectiveness of any compression algorithm depends on the exact nature of the data being compressed. Data files that are already compressed, such as compressed picture formats or rpm files, usually lead to a low compression ratio.

It is good practice to use a single top-level directory, which can contain other directories and files, to simplify extraction of the files in an organized way.

To create a compressed tar archive, one of the following **tar** options can be specified:

- **z** for gzip compression (filename.tar.gz or filename.tgz)
- **j** for bzip2 compression (filename.tar.bz2)
- **J** for xz compression (filename.tar.xz)

Create (c option) a gzip-compressed (z option) tar archive **/root/etcbackup.tar.gz** of the **/etc** directory on serverX:

```
[root@serverX ~]$ tar czf /root/etcbackup.tar.gz /etc
```

Create (c option) a bzip2-compressed (j option) tar archive **/root/logbackup.tar.bz2** of the **/var/log** directory on serverX:

```
[root@serverX ~]$ tar cjf /root/logbackup.tar.bz2 /var/log
```

Create (c option) a xz-compressed (J option) tar archive **/root/sshconfig.tar.bz2** of the **/etc/ssh** directory on serverX:

```
[root@serverX ~]$ tar cJf /root/sshconfig.tar.xz /etc/ssh
```

Extract a compressed tar archive

The first step when extracting a compressed tar archive is to determine where the archived files should be extracted to, then create and change to the target directory. To successfully extract the archive, it is usually not necessary to use the same compression option used when creating the archive, as the **tar** command will determine which compression was used. It is valid to add the decompression method to the **tar** options as follows:

Extract (x option) the contents of a gzip-compressed (z option) tar archive named **/root/etcbackup.tar.gz** to the directory **/tmp/etcbackup**:

```
[root@serverX ~]$ mkdir /tmp/etcbackup
[root@serverX ~]$ cd /tmp/etcbackup
[root@serverX etcbackup]$ tar xzf /root/etcbackup.tar.gz
```

Extract (x option) the contents of a bzip2-compressed (j option) tar archive named **/root/logbackup.tar.bz2** to the directory **/tmp/logbackup**:

```
[root@serverX ~]$ mkdir /tmp/logbackup
[root@serverX ~]$ cd /tmp/logbackup
[root@serverX logbackup]$ tar xjf /root/logbackup.tar.bz2
```

Extract (x option) the contents of a xz-compressed (J option) tar archive named **/root/sshbackup.tar.xz** to the directory **/tmp/sshbackup**:

```
[root@serverX ~]$ mkdir /tmp/sshbackup
[root@serverX ~]$ cd /tmp/sshbackup
[root@serverX sshbackup]$ tar xJf /root/sshbackup.tar.xz
```



Note

Listing a compressed **tar** archive works in the same way as listing an uncompressed **tar** archive.



Note

Additionally, **gzip**, **bzip2**, and **xz** can be used independently to compress single files. For example, **gzip etc.tar** results in the compressed file **etc.tar.gz**, while **bzip2 abc.tar** results in the compressed file **abc.tar.bz2** and **xz myarchive.tar** results in the compressed file **myarchive.tar.xz**.

The corresponding decompress commands are **gunzip**, **bunzip2**, and **unxz**. For example, **gunzip /tmp/etc.tar.gz** results in the uncompressed tar file **etc.tar**, while **bunzip2 abc.tar.bz2** results in the uncompressed tar file **abc.tar** and **unxz myarchive.tar.xz** results in the uncompressed tar file **myarchive.tar**.

Overview of tar options

The **tar** command has many options to use. The following table lists some common options and their meanings.

Overview of tar options

Option	Meaning
c	Create a new archive.
x	Extract from an existing archive.
t	List the contents of an archive.
v	Verbose; shows which files get archived or extracted.
f	File name; this option needs to be followed by the file name of the archive to use/create.
p	Preserve the permissions of files and directories when extracting an archive, without subtracting the umask.
z	Use gzip compression (.tar.gz).
j	Use bzip2 compression (.tar.bz2). bzip2 typically achieves a better compression ratio than gzip .
J	Use xz compression (.tar.xz). xz typically achieves a better compression ratio than bzip2 .

R

References

tar(1), **gzip(1)**, **gunzip(1)**, **bzip2(1)**, **bunzip2(1)**, **xz(1)**, **unxz(1)** man pages

Practice: Backing Up and Restoring Files From a tar Archive

Guided exercise

In this lab, students will create and extract archives with **tar**.

Outcomes:

Students will back up a directory tree and extract the archive content to another location.

- 1. Since only the root user can read all the contents of the **/etc** directory, we are going to back up the directory and log into serverX as **root**.

```
[student@desktopX ~]$ ssh root@serverX
```

- 2. Create an archive of **/etc** using **gzip** compression to back up the configuration file directory **/etc**. Save the file as **/tmp/etc.tar.gz**.

```
[root@serverX ~]# tar czf /tmp/etc.tar.gz /etc
```

- 3. Verify that the backup file **etc.tar.gz** is a valid archive by decompressing the file to a newly created directory named **/backuptest** on serverX.
 - 3.1. Create the target directory **/backuptest**.

```
[root@serverX ~]# mkdir /backuptest
```

- 3.2. Switch to the **/backuptest** directory, where we will extract the files from the **etc.tar.gz** archive.

```
[root@serverX ~]# cd /backuptest
```

- 3.3. Extract the **etc.tar.gz** archive to the **/backuptest** directory.

```
[root@serverX backuptest]# tar xzf /tmp/etc.tar.gz
```

Copying Files Between Systems Securely

Objectives

After completing this section, students should be able to copy files securely to or from a remote system running sshd.

Copy files to or from a remote location with scp

The **ssh** command is useful for securely running shell commands on remote systems. It can also be used to securely copy files from one machine to another. The **scp** command transfers files from a remote host to the local system or from the local system to a remote host. It utilizes the SSH server for authentication and encrypted data transfer.

Remote file system locations are always specified in the format **[user@]host :/path** for either the source or target location of the files to be transferred. The **user@** portion is optional and, if it is missing, the current local user that invokes the **scp** command is used. Before the transfer is initiated, the user must authenticate with the SSH server by password or SSH keys.

This example shows how to copy the local files on desktopX, **/etc/yum.conf** and **/etc/hosts**, securely to the account student on the remote system serverX into the directory **/home/student/**:

```
[student@desktopX ~]$ scp /etc/yum.conf /etc/hosts serverX:/home/student  
student@serverX's password: student  
      yum.conf          100%   813      0.8KB/s  00:00  
      hosts            100%   227      0.2KB/s  00:00
```

A user can copy a file from a remote account on a remote machine to the local file system with **scp**. In this example, copy the file **/etc/hostname** from the account student on the serverX machine to the local directory **/home/student/**.

```
[student@desktopX ~]$ scp serverX:/etc/hostname /home/student/  
student@serverX's password: student  
      hostname         100%    22      0.0KB/s  00:00
```

To copy a whole directory tree recursively, the **-r** option is available. In the following example, the remote directory **/var/log** on serverX is copied recursively to the local directory **/tmp/** on desktopX. To be able to read all the contents of the **/etc** directory, the root user must connect to the remote location.

```
[student@desktopX ~]$ scp -r root@serverX:/var/log /tmp  
root@serverX's password: redhat  
...
```

Transfer files remotely with sftp

If an interactive tool is preferred when uploading or downloading files to a SSH server, the **sftp** command can be used. A session with **sftp** is similar to a classic FTP session, but uses the secure authentication mechanism and encrypted data transfer of the SSH server.

To initiate a **sftp** session, the **sftp** expects a remote location in the format **[user@]host**, where the **user@** portion is optional and, if it is missing, the user invoking the **sftp** command

is used. To establish the **sftp** session, authenticating with any of the methods the SSH server accepts is necessary.

```
[student@desktopX ~]$ sftp serverX
student@serverX's password: student
Connected to serverX.
sftp>
```

The **sftp** session accepts various commands that work the same way on the remote file system as they do in the local file system, such as **ls**, **cd**, **mkdir**, **rmdir**, and **pwd**. In addition, there are the **put** and **get** commands for uploading and downloading files. The **exit** command exits the **sftp** session.

Upload the local file **/etc/hosts** to the newly created directory **/home/student/hostbackup** on the remote host serverX. The **sftp** session always assumes that the **put** command is followed by a file on the local file system and starts in the connecting user's home directory; in this case, **/home/student**:

```
sftp> mkdir hostbackup
sftp> cd hostbackup
sftp> put /etc/hosts
Uploading /etc/hosts to /home/student/hostbackup/hosts
/etc/hosts                                         100%  227      0.2KB/s  00:00
sftp>
```

To download the remote file **/etc/yum.conf** from the remote host to the current directory on the local file system, execute the command **get /etc/yum.conf** and exit the **sftp** session with the **exit** command.

```
sftp> get /etc/yum.conf
Fetching /etc/yum.conf to yum.conf
/etc/yum.conf                                         100%  813      0.8KB/s  00:00
sftp> exit
[student@desktopX ~]$
```

R

References

scp(1) and **sftp(1)** man pages

Practice: Copying Files Over the Network With **scp**

Guided exercise

In this lab, students will copy files from a remote system to a local directory by using **scp**.

Outcomes:

Students will copy files from a remote host to a directory on the local machine.

- 1. Remotely copy the **/etc/ssh** directory on the serverX machine to the newly created directory **/home/student/serverbackup** on desktopX by using **scp**.

- 1.1. Create the target directory **/home/student/serverbackup** on desktopX.

```
[student@desktopX ~]$ mkdir /home/student/serverbackup
```

- 1.2. Recursively copy the directory **/etc/ssh** from serverX to the **/home/student/serverbackup** directory on desktopX with the **scp** command. Note that only the root user can read all the content in the **/etc/ssh** directory.

```
[student@desktopX ~]$ scp -r root@serverX:/etc/ssh /home/student/serverbackup
```

Synchronizing Files Between Systems Securely

Objectives

After completing this section, students should be able to efficiently and securely synchronize the contents of a local file or directory with a remote copy.

Synchronize files and folders with **rsync**

The **rsync** tool is another way to securely copy files from one system to another. It differs from **scp** in that if two files or directories are similar between two systems, **rsync** only needs to copy the differences between the systems, while **scp** would need to copy everything.

One of the advantages of **rsync** is that it can copy files between a local system and a remote system securely and efficiently. While the initial synchronization of a directory takes about the same time as copying it, any subsequent synchronization only requires the differences to be copied over the network.

One of the most important options of **rsync** is the **-n** option to perform a dry run. A dry run is a simulation of what happens when the command really gets executed. It will display the changes it will perform when the command is executed without the dry run option. It is recommended to perform a dry run of any **rsync** operation to ensure no important files get overwritten or deleted.

The two most common options when synchronizing files and folders with **rsync** are the **-a** and **-v** options. While the **-v** option adds verbosity to the output as the synchronization proceeds, the **-a** option stands for "archive mode" and enables the following options all in one:

- **-r**, synchronize recursively the whole directory tree
- **-l**, synchronize symbolic links
- **-p**, preserve permissions
- **-t**, preserve time stamps
- **-g**, preserve group ownership
- **-o**, preserve the owner of the files
- **-D**, synchronize device files

While the **-a** already synchronizes symbolic links, there are additional options necessary to preserve hardlinks, as they are treated as separate files instead. The **-H** option enables the handling of hardlinks, so the **rsync** command will identify the hardlinks present in the source folder and link the files accordingly in the destination folder instead of just copying them as separate files.



Note

The **-a** option does not synchronize advanced file permissions, such as ACLs or SELinux file contexts. To enable the synchronization of ACLs, the **-A** option is required in addition to the **-a** option, while to synchronize the SELinux contexts from the source files to the target files, the **-X** option is to be added.

The basic way of using **rsync** is to synchronize two local folders. In the following example, the **/var/log** directory gets a synchronized copy in the **/tmp** folder. The **log** directory with its content is created in the **/tmp** directory.

```
[student@desktopX ~]$ su -
Password: redhat
[root@desktopX ~]# rsync -av /var/log /tmp
...
```

To only synchronize the content of a folder without newly creating the folder in the target directory, a trailing slash needs to be added at the end of the source directory. In this example, the **log** directory is not created in the **/tmp** folder. Only the content of the **/var/log/** directory is synchronized into the **/tmp** folder.

```
[root@desktopX ~]# rsync -av /var/log/ /tmp
...
```



Important

When entering the source directory for **rsync**, it is very important to remember that whether a trailing slash is present on the directory name matters. It will determine whether the *directory* or just the *contents of the directory* are synchronized to the target. Note: Tab-completion will automatically add a trailing slash to the end of directory names.

Similar to **scp**, the **rsync** command expects remote file system locations to be specified in the format **[user@]host:/path**. In case the optional **user@** portion is missing, the user invoking the **rsync** command is used to connect to the remote location. It is possible to use a remote location as either source or target. In the following example, the local folder **/var/log** gets a synchronized copy in the **/tmp** directory on the serverX machine. For **rsync** to synchronize ownership of the transferred files, the target location must be written as user root, so connect to the remote system serverX as the root user. The connecting user root must then authenticate with the SSH server by any of the accepted methods; for example, password or SSH keys.

```
[root@desktopX ~]$ rsync -av /var/log serverX:/tmp
root@serverX's password: redhat
...
```

In the same way, the remote folder **/var/log** on serverX can be synchronized to the local directory **/tmp** on desktopX:

```
[root@desktopX ~]$ rsync -av serverX:/var/log /tmp
```

```
root@serverX's password: redhat  
...
```



References

rsync(1) man page

Practice: Synchronizing Two Directories Securely With **rsync**

Guided exercise

In this lab, students will synchronize a folder with a remote system by using **rsync**.

Outcomes:

A directory will be synchronized from a remote machine to the local machine. After files on the remote machine have changed, it will be synchronized with the local machine again and only the modifications will be transferred. Both systems will end with identical content in the directories synchronized with **rsync**.

- 1. Securely create an initial copy of the **/var/log** directory tree on serverX to a newly created directory named **/serverlogs** on desktopX with the **rsync** command.

- 1.1. In order to create the target directory **/serverlogs**, switch to the root user account with the **su** command.

```
[student@desktopX ~]$ su -  
Password: redhat  
[root@desktopX ~]#
```

- 1.2. Create the target directory **/serverlogs** on desktopX where the log files of serverX will be synchronized.

```
[root@desktopX ~]# mkdir /serverlogs
```

- 1.3. Use the **rsync** command to synchronize the **/var/log** directory tree on serverX to the **/serverlogs** directory on desktopX. Note that only the root user can read all the content in the **/var/log** directory on serverX. All files will be transferred in the initial synchronization.

```
[root@desktopX ~]# rsync -av root@serverX:/var/log /serverlogs  
...
```

- 2. As root on serverX, run **logger "Log files synchronized"** to get a new entry in the log file **/var/log/messages** to reflect when the last synchronization took place.

```
[root@desktopX ~]# ssh root@serverX 'logger "Log files synchronized"'  
Password: redhat  
[root@desktopX ~]#
```

- 3. Securely synchronize the **/var/log** directory tree on serverX to the **/serverlogs** directory on desktopX with the **rsync** command again. Note that this time only the changed log file(s) will be transferred.

```
[root@desktopX ~]# rsync -av root@serverX:/var/log /serverlogs
```

...

Lab: Archiving and Copying Files Between Systems

Performance checklist

In this lab, students will use **rsync**, **scp**, and **tar** to archive and back up folder contents.

Outcomes:

Students will synchronize a remote folder to a local directory; an archive is then created with the synchronized folder as content; the archive gets copied to the remote machine and extracted to a newly created directory.

Before you begin...

Reset your serverX system.

1. Synchronize the **/etc** directory tree on serverX to the **/configsync** directory on desktopX.
2. On desktopX, create an archive named **/root/configfile-backup-serverX.tar.gz** with the **/configsync** directory as content, and copy the archive to the **/root** directory on serverX for backup purposes with the **scp** command.
3. To prepare the archived directory tree for comparison with the currently actively used configuration files on serverX, extract the contents of the **/root/configfile-backup-serverX.tar.gz** archive to the **/tmp/savedconfig/** directory on serverX.

Solution

In this lab, students will use **rsync**, **scp**, and **tar** to archive and back up folder contents.

Outcomes:

Students will synchronize a remote folder to a local directory; an archive is then created with the synchronized folder as content; the archive gets copied to the remote machine and extracted to a newly created directory.

Before you begin...

Reset your serverX system.

1. Synchronize the **/etc** directory tree on serverX to the **/configsync** directory on desktopX.
 - 1.1. To be able to create the target directory **/configsync**, switch to the root user account using **su**.

```
[student@desktopX ~]$ su -  
Password: redhat  
[root@desktopX ~]#
```

- 1.2. Create the target directory for the configuration files on desktopX.

```
[root@desktopX ~]# mkdir /configsync
```

- 1.3. Use the **rsync** command to synchronize the **/etc** directory tree on serverX to the **/configsync** directory on desktopX. Be aware that only the root user can read all the content in the **/etc** directory on serverX.

```
[root@desktopX ~]# rsync -av root@serverX:/etc /configsync  
...
```

2. On desktopX, create an archive named **/root/configfile-backup-serverX.tar.gz** with the **/configsync** directory as content, and copy the archive to the **/root** directory on serverX for backup purposes with the **scp** command.

- 2.1. Store the **/configsync** directory in the **/root/configfile-backup-serverX.tar.gz** archive.

```
[root@desktopX ~]# tar czf /root/configfile-backup-serverX.tar.gz /configsync
```

- 2.2. Create a backup copy of the **/root/configfile-backup-serverX.tar.gz** on serverX

```
[root@desktopX configsync]# scp /root/configfile-backup-serverX.tar.gz  
root@serverX:/root  
Password: redhat  
...
```

3. To prepare the archived directory tree for comparison with the currently actively used configuration files on serverX, extract the contents of the **/root/configfile-backup-serverX.tar.gz** archive to the **/tmp/savedconfig/** directory on serverX.

- 3.1. Connect to the serverX machine as root by using **ssh**.

```
[root@desktopX configsync]# ssh root@serverX  
Password: redhat  
[root@serverX ~]#
```

- 3.2. Create the target directory **/tmp/savedconfig/**, where the contents of the **/root/configfile-backup-serverX.tar.gz** archive will be extracted.

```
[root@serverX ~]# mkdir /tmp/savedconfig
```

- 3.3. Change to the target directory **/tmp/savedconfig/** on serverX.

```
[root@serverX ~]# cd /tmp/savedconfig  
[root@serverX savedconfig]#
```

- 3.4. Extract the contents of the **/root/configfile-backup-serverX.tar.gz** archive to the **/tmp/savedconfig/** directory on serverX.

```
[root@serverX savedconfig]# tar xzf /root/configfile-backup-serverX.tar.gz
```

Summary

Managing Compressed tar Archives

The **tar** command provides a set of different compression methods to archive files and restore them from an archive.

Copying Files Between Systems Securely

Besides providing a secure remote shell, the ssh service also provides the **scp** and **sftp** as secure ways to transfer files from and to a remote system running the SSH server.

Synchronizing Files Between Systems Securely

The **rsync** command securely and efficiently synchronizes files with a remote location.



CHAPTER 13

INSTALLING AND UPDATING SOFTWARE PACKAGES

Overview	
Goal	To download, install, update, and manage software packages from Red Hat and YUM package repositories.
Objectives	<ul style="list-style-type: none">Register systems with your Red Hat account and entitle them to software updates for installed products.Explain what an RPM package is and how RPM packages are used to manage software on a Red Hat Enterprise Linux system.Find, install, and update software packages using the <code>yum</code> command.Enable and disable use of Red Hat or third-party YUM repositories.Examine and install downloaded software package files.
Sections	<ul style="list-style-type: none">Attaching Systems to Subscriptions for Software Updates (and Practice)RPM Software Packages and YUM (and Practice)Managing Software Updates with <code>yum</code> (and Practice)Enabling <code>yum</code> Software Repositories (and Practice)Examining RPM Package Files (and Practice)
Lab	Installing and Updating Software Packages

Attaching Systems to Subscriptions for Software Updates

Objectives

Register systems with your Red Hat account and entitle them to software updates for installed products.

Red Hat Subscription Management

Red Hat Subscription Management provides tools that can be used to entitle machines to product subscriptions, allowing administrators to get updates to software packages and track information about support contracts and subscriptions used by the systems. Standard tools such as `PackageKit` and `yum` can obtain software packages and updates through a content distribution network provided by Red Hat.

There are four basic tasks performed with Red Hat Subscription Management tools:

- **Register** a system to associate that system to a Red Hat account. This allows Subscription Manager to uniquely inventory the system. When no longer in use, a system may be unregistered.
- **Subscribe** a system to entitle it to updates for selected Red Hat products. Subscriptions have specific levels of support, expiration dates, and default repositories. The tools can be used to either auto-attach or select a specific entitlement. As needs change, subscriptions may be removed.
- **Enable repositories** to provide software packages. Multiple repositories are enabled by default with each subscription, but other repositories such as updates or source code can be enabled or disabled as needed.
- **Review and track** entitlements which are available or consumed. Subscription information can be viewed locally on a specific system or, for an account, in either the Red Hat Customer Portal **Subscriptions** page or the Subscription Asset Manager (SAM).

Register a system

To register a system with the subscription management service, launch **subscription-manager-gui** by selecting **Applications > Other > Red Hat Subscription Manager** from the main GNOME menu. Enter the password for *root* when prompted to authenticate. This will display the following **Subscription Manager** window.

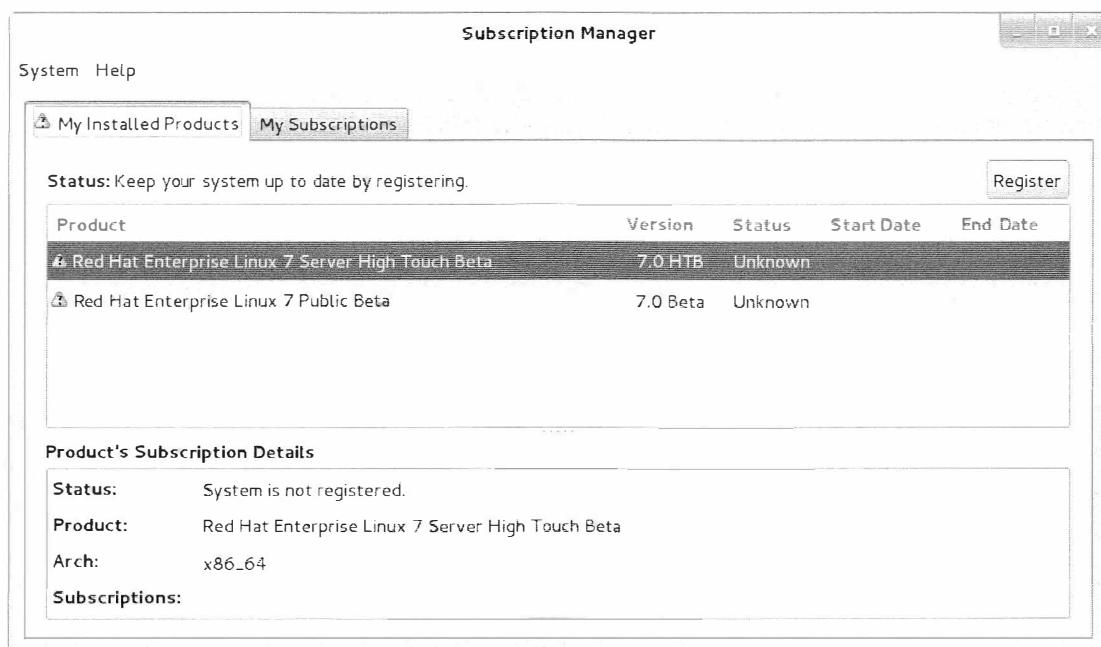


Figure 13.1: The main window of Red Hat Subscription Manager

To register the system, click the **Register** button in the top-right corner of the **Subscription Manager** window. This will bring up the following dialog:

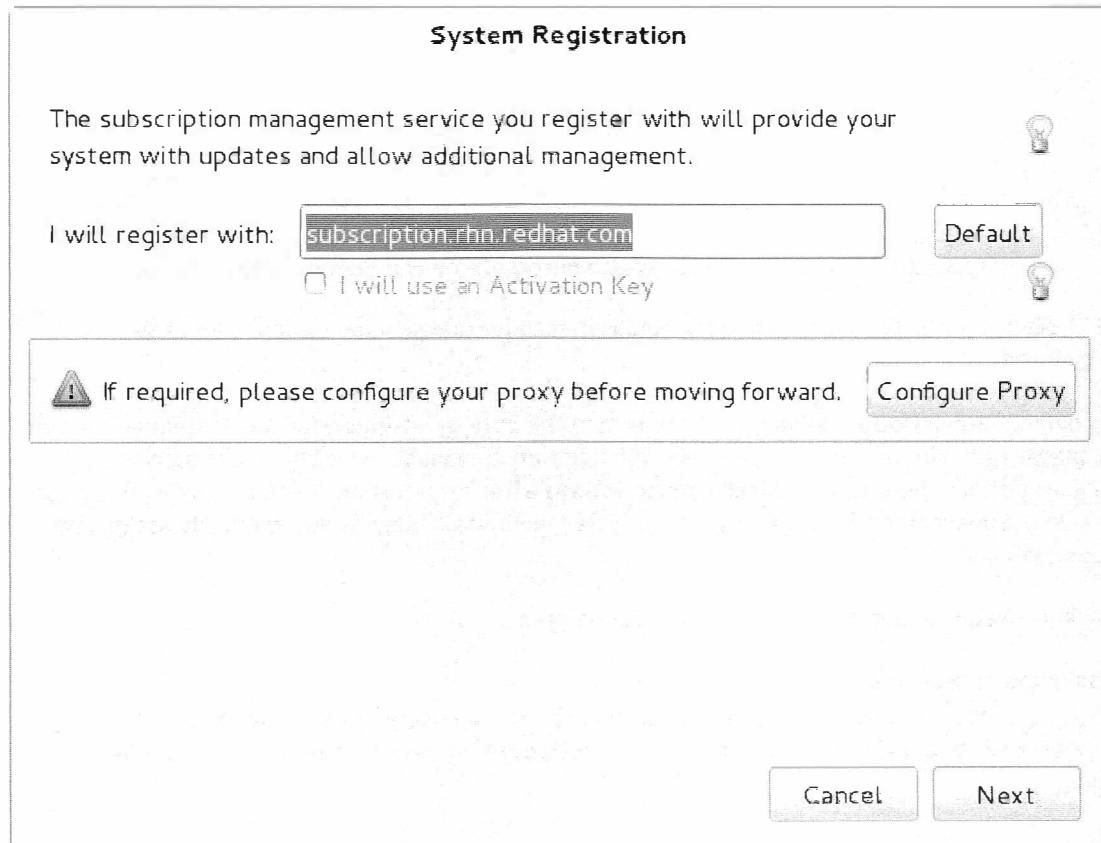


Figure 13.2: The service location dialog of Red Hat Subscription Manager

This dialog box registers a system with a subscription server. The default (subscription.rhn.redhat.com) will register the server to Red Hat's "hosted" content distribution network.

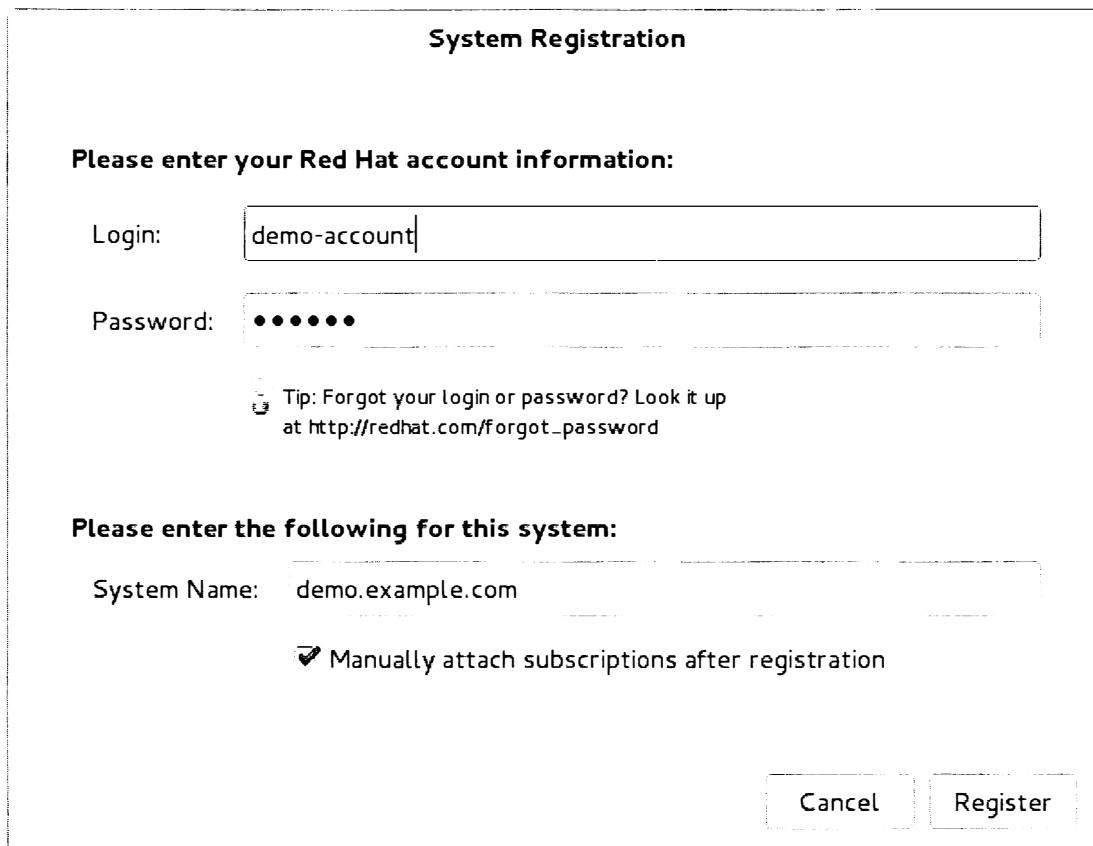


Figure 13.3: The account information dialog of Red Hat Subscription Manager

Click **Next**, then authenticate using the Red Hat account to which the system should be registered.

By default, **Subscription Manager** will try to find the best subscription for this system out of all available subscriptions. If more than one subscription is available, or a specific subscription is required, check the **Manually attach subscriptions after registration** checkbox. With this option checked, **Subscription Manager** will only register the system and not automatically assign any subscriptions.

Click the **Register** button to complete the registration.

Assigning subscriptions

To assign subscriptions to a system, navigate to the **All Available Subscriptions** tab in the main window of **Subscription Manager**, then click the **Update** button to retrieve a list of available subscriptions.



Figure 13.4: All Available Subscriptions tab of Red Hat Subscription Manager

From this list, select one or more subscriptions to assign to this system, then click the **Attach** button.

If there is more than one contract for a specific subscription, a new dialog will open up, asking you to select which contract to use. Note that there are different contracts for *Physical* and *Virtual* systems.

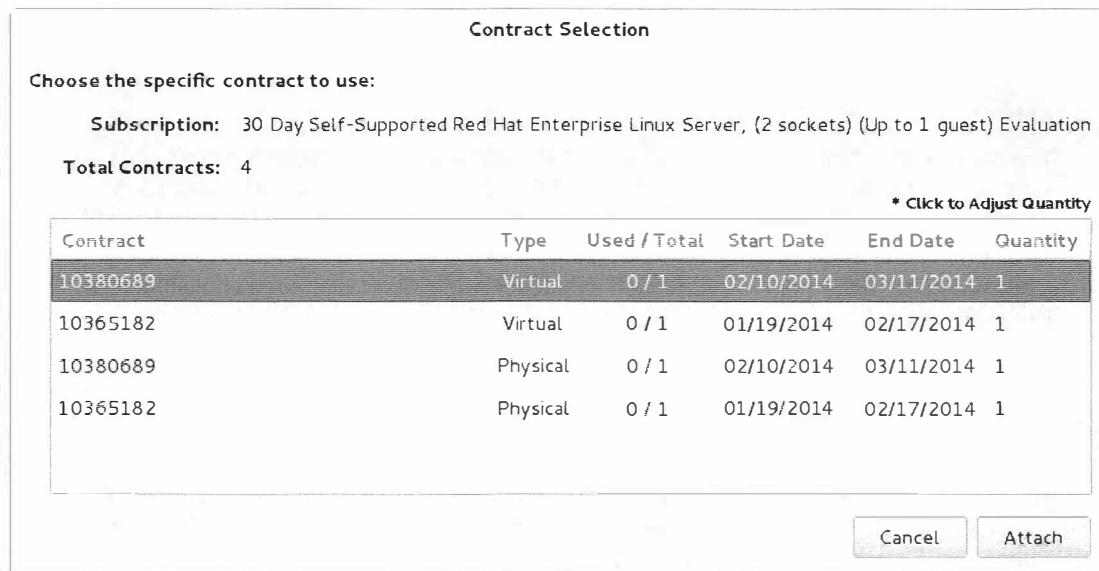


Figure 13.5: Contract selection dialog of Red Hat Subscription Manager

After a subscription has been assigned, close the **Subscription Manager** window. The system is now properly subscribed and ready to receive updates and/or install new software from Red Hat.

Automating registrations and subscriptions

Use **subscription-manager(8)** to register a system without using a graphical environment. The **subscription-manager** command can automatically attach a system to the best-matched compatible subscriptions for the system.

- Register a system to a Red Hat account:

```
[root:@serverX ~]# subscription-manager register --username=yourusername --password=yourpassword
```

- View available subscriptions:

```
[root:@serverX ~]# subscription-manager list --available | less
```

- Auto-attach a subscription:

```
[root:@serverX ~]# subscription-manager attach --auto
```

- View consumed subscriptions:

```
[root:@serverX ~]# subscription-manager list --consumed
```

- Unregister a system:

```
[root:@serverX ~]# subscription-manager unregister
```



Note

subscription-manager can also be used in conjunction with *activation keys*, allowing registration and assignment of predefined subscriptions, without using a username or password. This method of registration can be very useful for automated installations and deployments. Activation keys are usually issued by an on-premise subscription management service, such as Subscription Asset Manager, and will not be discussed in detail in this course.

Entitlement certificates

An entitlement is a subscription that's been attached to a system. Digital certificates are used to store current information about entitlements on the local system. Once registered, the entitlement certificates are stored in **/etc/pki** and its subdirectories.

- **/etc/pki/product** contains certificates which indicate Red Hat products installed on the system.
- **/etc/pki/consumer** contains certificates which indicate the Red Hat account to which the system is registered.

- **/etc/pki/entitlement** contains certificates which indicate which subscriptions are attached to the system.

The certificates can be inspected with the **rct** utility directly, but normally the **subscription-manager** tools are a more user-friendly way to examine the subscriptions that are attached to the system.



Important

Older versions of Red Hat Enterprise Linux originally supported a different subscription management method, *RHN Classic*. RHN Classic is not supported by Red Hat Enterprise Linux 7.

The method covered in this section, *Red Hat Subscription Management*, is the only one used by RHEL 7, and is the default method used by RHEL 6 after RHEL 6.3, and RHEL 5 after RHEL 5.9. RHEL 4 only supports the old method. More information on both methods is available in the references at the end of this section.



References

subscription-manager -gui(8), **subscription-manager(8)**, and **rct(8)** man pages

Get started with Red Hat Subscription Management
<https://access.redhat.com/site/articles/433903>

Red Hat Subscription Management: Migrating from RHN and Satellite
https://access.redhat.com/site/documentation/en-US/Red_Hat_Subscription_Management/1/html-single/MigratingRHN/

Practice: Red Hat Subscription Management

Quiz

Match the following items to their counterparts in the table.

Enable repositories	Register	Review and track	Subscribe
---------------------	----------	------------------	-----------

Description	Task
Determine the number of available subscriptions	
Enable a system to use selected Red Hat products	
Attach a system to a Red Hat account	
Provide software packages	

Solution

Match the following items to their counterparts in the table.

Description	Task
Determine the number of available subscriptions	Review and track
Enable a system to use selected Red Hat products	Subscribe
Attach a system to a Red Hat account	Register
Provide software packages	Enable repositories

RPM Software Packages and Yum

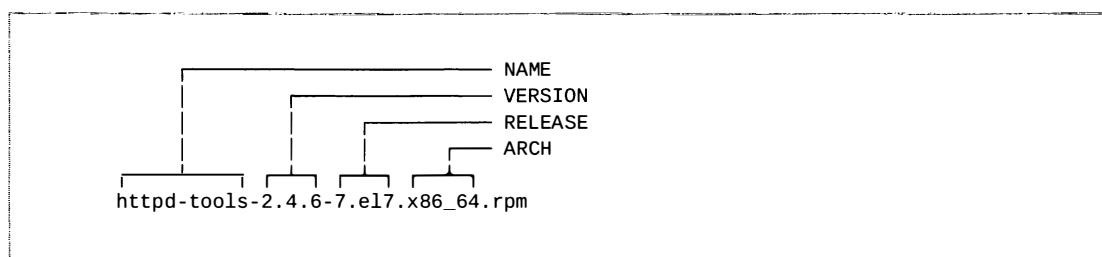
Objectives

After completing this section, students should be able to explain what an RPM package is and how RPM packages are used to manage software on a Red Hat Enterprise Linux system.

Software packages and RPM

Many years ago, Red Hat developed the **RPM Package Manager**, which provides a standard way to package software for distribution. Managing software in the form of *RPM packages* is much simpler than working with software that has simply been extracted into a file system from an archive. It lets administrators track which files were installed by the software package and which ones need to be removed if it is uninstalled, and check to ensure supporting packages are present when it is installed. Information about installed packages is stored in a local RPM database on each system. All software provided by Red Hat for Red Hat Enterprise Linux is provided as an RPM package.

RPM package files are named using a combination of the package **name-version-release.architecture**:



- NAME is one or more words describing the contents (httpd-tools).
- VERSION is the version number of the original software (2.4.6).
- RELEASE is the release number of the package based on that version, and is set by the packager, who might not be the original software developer (7.el7).
- ARCH is the processor architecture the package was compiled to run on. "noarch" indicates that this package's contents are not architecture-specific (x86_64).

When installing packages from repositories, only the package name is required. The package with the higher version will be installed. If there are multiple files with the same version, the package with the higher release number will be installed.

Each RPM package is a special archive made up of three components:

- The files installed by the package.
- Information about the package (metadata), such as the name/version/release/arch; a summary and description of the package; whether it requires other packages to be installed; licensing; a package changelog; and other details.
- Scripts which may run when this package is installed, updated, or removed, or which are triggered when other packages are installed, updated, or removed.

RPM packages can be digitally signed by the organization that packaged them. All packages from a particular source are normally signed by the same GPG private key. If the package is altered or corrupted, the signature will no longer be valid. This allows the system to verify package integrity before installing them. All RPM packages released by Red Hat are digitally signed.

Updates and Patches

When the upstream source code for a software package is patched by Red Hat, a complete RPM package is generated. If a package is newly added to a system, only the latest version of that package is needed, not every version of the package since the first release. For systems that need updating, the old version of the package is actually removed and the new version is installed. Configuration files are usually retained during an upgrade, but the exact behavior for a particular package is defined when the new version of the package is created.

In most cases, only one version or release of a package may be installed at a time. Typically, the RPM installation process will not allow files to be overwritten. If a package is built so that there are no conflicting filenames, then multiple versions may be installed. This is the case for the **kernel** package. Since a new kernel can only be tested by booting to that kernel, the package is specifically designed so that multiple versions may be installed at once. If the new kernel fails to boot, the old kernel is still available.

The yum package manager

Once a system is installed, additional software packages and updates are normally installed from a network *package repository*, most frequently through the Red Hat subscription management service discussed in the previous section. The **rpm** command may be used to install, update, remove, and query RPM packages. However, it does not resolve dependencies automatically and all packages must be listed. Tools such as **PackageKit** and **yum** are front-end applications for **rpm** and can be used to install individual packages or *package collections* (sometimes called *package groups*).

The **yum** command searches numerous repositories for packages and their dependencies so they may be installed together in an effort to alleviate dependency issues. The main configuration file for **yum** is **/etc/yum.conf** with additional repository configuration files located in the **/etc/yum.repos.d** directory. Repository configuration files include, at a minimum, a repo id (in square brackets), a name and the URL location of the package repository. The URL can point to a local directory (file) or remote network share (http, ftp, etc.). If the URL is pasted in a browser, the contents should display the RPM packages, possibly in one or more subdirectories, and a **repodata** directory with information about available packages.

The **yum** command is used to list repositories, packages, and package groups:

```
[root@serverX ~]# yum repolist
Loaded plugins: langpacks
repo id          repo name                                status
!rhel_dvd        Remote classroom copy of dvd           4,529
repolist: 4,529
[root@serverX ~]# yum list yum*
Loaded plugins: langpacks
Installed Packages
yum.noarch        3.4.3-118.2.el7                      @anaconda/7.0
yum-langpacks.noarch 0.4.2-3.el7                        @rhel_dvd
yum-metadata-parser.x86_64 1.1.4-10.el7                   @anaconda/7.0
yum-rhn-plugin.noarch 2.0.1-4.el7                        @rhel_dvd
yum-utils.noarch   1.1.31-24.el7                     @rhel_dvd
Available Packages
```

```
yum-plugin-aliases.noarch      1.1.31-24.el7      rhel_dvd
yum-plugin-changelog.noarch    1.1.31-24.el7      rhel_dvd
yum-plugin-tmprepo.noarch     1.1.31-24.el7      rhel_dvd
yum-plugin-verify.noarch      1.1.31-24.el7      rhel_dvd
yum-plugin-versionlock.noarch 1.1.31-24.el7      rhel_dvd
[root@serverX ~]# yum list installed
Loaded plugins: langpacks
Installed Packages
GConf2.x86_64                  3.2.6-8.el7        @rhel_dvd
ModemManager.x86_64             1.1.0-6.git20130913.el7  @rhel_dvd
ModemManager-glib.x86_64        1.1.0-6.git20130913.el7  @rhel_dvd
...
[root@serverX ~]# yum grouplist
...
Installed groups:
  Base
  Desktop Debugging and Performance Tools
  Dial-up Networking Support
  Fonts
  Input Methods
...
```

References

yum(8), yum.conf(5), rpm(8), rpm2cpio(8), and rpmkeys(8) man pages

Practice: RPM Software Packages

Quiz

Match the following items to their counterparts in the table.

Architecture	Changelog	GPG signature	Release
Repository	Version		

Description	Term
The upstream source code version	
List of reasons for each package build	
The version of the package build	
The processor type required for a specific package	
A collection of RPM packages and package groups	
Used to verify the source and integrity of a package	

Solution

Match the following items to their counterparts in the table.

Description	Term
The upstream source code version	Version
List of reasons for each package build	Changelog
The version of the package build	Release
The processor type required for a specific package	Architecture
A collection of RPM packages and package groups	Repository
Used to verify the source and integrity of a package	GPG signature

Managing Software Updates with yum

Objectives

After completing this section, students should be able to find, install, and update software packages using the **yum** command.

Working with yum

yum is a powerful command-line tool that can be used to more flexibly manage (install, update, remove, and query) software packages. Official Red Hat packages are normally downloaded from Red Hat's content distribution network. Registering a system to the subscription management service automatically configures access to software repositories based on the attached subscriptions.

Finding software with yum

- **yum help** will display usage information.
- **yum list** displays installed and available packages.

```
[root@serverX ~]# yum list 'http*'
Loaded plugins: langpacks
Available Packages
httpcomponents-client.noarch          4.2.5-4.el7      rhel_dvd
httpcomponents-core.noarch            4.2.4-6.el7      rhel_dvd
httpd.x86_64                          2.4.6-17.el7    rhel_dvd
httpd-devel.x86_64                    2.4.6-17.el7    rhel_dvd
httpd-manual.noarch                  2.4.6-17.el7    rhel_dvd
httpd-tools.x86_64                   2.4.6-17.el7    rhel_dvd
```

- **yum search KEYWORD** lists packages by keywords found in the name and summary fields only.

To search for packages that have "web server" in their name, summary, and description fields, use **search all**:

```
[root@serverX ~]# yum search all 'web server'
Loaded plugins: langpacks
=====
Matched: web server =====
freeradius.x86_64 : High-performance and highly configurable free RADIUS server
hsqldb.noarch : HypersQL Database Engine
httpd.x86_64 : Apache HTTP Server
libcurl.i686 : A library for getting files from web servers
libcurl.x86_64 : A library for getting files from web servers
mod_revocator.x86_64 : CRL retrieval module for the Apache HTTP server
mod_security.x86_64 : Security module for the Apache HTTP Server
python-paste.noarch : Tools for using a Web Server Gateway Interface stack
```

- **yum info PACKAGE NAME** gives detailed information about a package, including the disk space needed for installation.

To get information on the Apache HTTP Server:

```
[root@serverX ~]# yum info httpd
```

```
Loaded plugins: langpacks
Available Packages
Name        : httpd
Arch       : x86_64
Version    : 2.4.6
Release   : 17.el7
Size      : 1.1 M
Repo      : rhel_dvd
Summary    : Apache HTTP Server
URL       : http://httpd.apache.org/
License    : ASL 2.0
Description: The Apache HTTP Server is a powerful, efficient, and extensible
             : web server.
```

- **yum provides PATHNAME** displays packages that match the pathname specified (which often include wildcard characters).

To find packages that provide the **/var/www/html** directory, use:

```
[root@serverX ~]# yum provides /var/www/html
Loaded plugins: langpacks
httpd-2.4.6-17.el7.x86_64 : Apache HTTP Server
Repo        : rhel_dvd
Matched from:
Filename   : /var/www/html

1:php-pear-1.9.4-21.el7.noarch : PHP Extension and Application Repository
                               : framework
Repo        : rhel_dvd
Matched from:
Filename   : /var/www/html
```

Installing and removing software with yum

- **yum install PACKAGE NAME** obtains and installs a software package, including any dependencies.

```
[root@serverX ~]# yum install httpd
Loaded plugins: langpacks
Resolving Dependencies
--> Running transaction check
--> Package httpd.x86_64 0:2.4.6-17.el7 will be installed
--> Processing Dependency: httpd-tools = 2.4.6-17.el7 for package:
    httpd-2.4.6-17.el7.x86_64
--> Processing Dependency: /etc/mime.types for package: httpd-2.4.6-17.el7.x86_64
--> Processing Dependency: libapr-1.so.0()(64bit) for package:
    httpd-2.4.6-17.el7.x86_64
--> Processing Dependency: libaprutil-1.so.0()(64bit) for package:
    httpd-2.4.6-17.el7.x86_64
--> Running transaction check
--> Package apr.x86_64 0:1.4.8-3.el7 will be installed
--> Package apr-util.x86_64 0:1.5.2-6.el7 will be installed
--> Package httpd-tools.x86_64 0:2.4.6-17.el7 will be installed
--> Package mailcap.noarch 0:2.1.41-2.el7 will be installed
--> Finished Dependency Resolution

Dependencies Resolved
```

```

=====
Package      Arch   Version       Repository    Size
=====
Installing:
httpd        x86_64  2.4.6-17.el7   rhel_dvd     1.1 M
Installing for dependencies:
apr          x86_64  1.4.8-3.el7    rhel_dvd     100 k
apr-util     x86_64  1.5.2-6.el7    rhel_dvd     90  k
httpd-tools  x86_64  2.4.6-17.el7   rhel_dvd     76  k
mailcap      noarch  2.1.41-2.el7   rhel_dvd     31  k

Transaction Summary
=====
Install 1 Package (+4 Dependent packages)

Total download size: 1.4 M
Installed size: 4.3 M
Is this ok [y/d/N]:
```

- **yum update PACKAGE NAME** obtains and installs a newer version of the software package, including any dependencies. Generally the process tries to preserve configuration files in place, but in some cases, they may be renamed if the packager thinks the old one will not work after the update. With no PACKAGE NAME specified, it will install all relevant updates.

```
[root@serverX ~]# yum update
```

Since a new kernel can only be tested by booting to that kernel, the package is specifically designed so that multiple versions may be installed at once. If the new kernel fails to boot, the old kernel is still available. Using **yum update kernel** will actually *install* the new kernel. The configuration files hold a list of packages to "always install" even if the administrator requests an update.



Note

Use **yum list kernel** to list all installed and available kernels. To view the currently running kernel, use the **uname** command. The **-r** option will show only the kernel version and release, and the **-a** option will show the kernel release and additional information.

```

[root@serverX ~]# yum list kernel
Loaded plugins: langpacks
Installed Packages
kernel.x86_64           3.10.0-123.0.1.el7          @anaconda/7.0
                                         rpms
[root@serverX ~]# uname -r
3.10.0-123.el7.x86_64
[root@serverX ~]# uname -a
Linux demo.example.com 3.10.0-123.el7.x86_64 #1 SMP Tue Nov 26 16:51:22 EST
2013 x86_64 x86_64 x86_64 GNU/Linux
```

- **yum remove PACKAGE NAME** removes an installed software package, including any supported packages.

```
[root@serverX ~]# yum remove httpd
```



Warning

yum remove will remove the package(s) listed and any package that requires the package(s) being removed (and package(s) which require those packages, and so on). This can lead to unexpected removal of packages, so carefully check the list of packages to be removed.

Installing and removing groups of software with yum

- **yum** also has the concept of *groups*, which are collections of related software installed together for a particular purpose. In Red Hat Enterprise Linux 7, there are two kinds of groups. Regular groups are collections of packages. *Environment groups* are collections of other groups which include their own packages. The packages or groups provided by a group may be *mandatory* (must be installed if the group is installed), *default* (are normally installed if the group is installed), or *optional* (are not installed when the group is unless asked for specifically).

Like **yum list**, the **yum group list** (or **yum grouplist**) command will show the names of installed and available groups. Some groups are normally installed through environment groups and are hidden by default. These hidden groups can also be listed with the **yum group list hidden** command. If the **ids** option is added, the group ID will also be shown. Groups can be installed, updated, removed, and otherwise queried by name or ID.

```
[root@serverX ~]# yum group list
Loaded plugins: langpacks
Available environment groups:
  Minimal install
  Infrastructure Server
  File and Print Server
  Web Server
  Virtualization Host
  Server with GUI
Installed groups:
  Base
  Desktop Debugging and Performance Tools
  Dial-up Networking Support
  Fonts
  Input Methods
  Internet Browser
  PostgreSQL Database server
  Printing client
  X Window System
Available Groups:
  Additional Development
  Backup Client
  Backup Server
  ...
```

- Information about a group is displayed with **yum group info** (or **yum groupinfo**). It includes a list of mandatory, default, and optional package names or group IDs. The package names or group IDs may have a marker in front of them.

Marker	Meaning
=	Package is installed, was installed as part of the group
+	Package isn't installed, will be if the group is installed or updated
-	Package isn't installed, will not be if the group is installed or updated
no marker	Package is installed, but was not installed through the group.

```
[root@serverX ~]# yum group info "Identity Management Server"
Loaded plugins: langpacks

Group: Identity Management Server
Group-Id: identity-management-server
Description: Centralized management of users, servers and authentication policies.
Default Packages:
+389-ds-base
+ipa-admin-tools
+ipa-server
+pki-ca
Optional Packages:
+ipa-server-trust-ad
+nuxwdog
+slapi-nis
```

- The **yum group install** (or **yum groupinstall**) command will install a group which will install its mandatory and default packages and the packages they depend on.

```
[root@serverX ~]# yum group install "Infiniband Support"
...
Transaction Summary
=====
Install 17 Packages (+7 Dependent packages)

Total download size: 9.0 M
Installed size: 33 M
Is this ok [y/d/N]:
...
```



Important

The behavior of **yum** groups has changed in Red Hat Enterprise Linux 7 from Red Hat Enterprise Linux 6 and earlier. In RHEL 7, groups are treated as *objects*, and are tracked by the system. If an installed group is updated, and new mandatory or default packages have been added to the group by the **yum** repository, those new packages will be installed on update.

RHEL 6 and earlier consider a group to be installed if all its mandatory packages have been installed; or if it had no mandatory packages, if any default or optional packages in the group are installed. In RHEL 7, a group is considered to be installed *only* if **yum group install** was used to install it. A new command in RHEL 7, **yum group mark install GROUPNAME** can be used to mark a group as installed, and any missing packages and their dependencies will be installed on the next update.

Finally, RHEL 6 and earlier did not have the two-word form of the **yum group** commands. In other words, in RHEL 6 the command **yum grouplist** existed, but the equivalent RHEL 7 command **yum group list** did not.

Viewing transaction history

- All install and remove transactions are logged in **/var/log/yum.log**.

```
[root@serverX ~]# tail -5 /var/log/yum.log
Feb 16 14:10:41 Installed: libnfs-1.1.3-5.el7.x86_64
Feb 16 14:10:42 Installed: libmthca-1.0.6-10.el7.x86_64
Feb 16 14:10:43 Installed: libmlx4-1.0.5-7.el7.x86_64
Feb 16 14:10:43 Installed: libibcm-1.0.5-8.el7.x86_64
Feb 16 14:10:45 Installed: rdma-7.0_3.13_rc8-3.el7.noarch
```

- A summary of install and remove transactions can be viewed with **yum history**.

```
[root@serverX ~]# yum history
Loaded plugins: langpacks
ID      | Login user           | Date and time   | Action(s)    | Altered
-----+-----+-----+-----+-----+-----+-----+-----+
 6 | Student User <student> | 2014-02-16 14:09 | Install      | 25
 5 | Student User <student> | 2014-02-16 14:01 | Install      | 1
 4 | System <unset>        | 2014-02-08 22:33 | Install      | 1112 EE
 3 | System <unset>        | 2013-12-16 13:13 | Erase       | 4
 2 | System <unset>        | 2013-12-16 13:13 | Erase       | 1
 1 | System <unset>        | 2013-12-16 13:08 | Install      | 266
history list
```

- A transaction can be reversed with the **history undo** options:

```
[root@serverX ~]# yum history undo 6
Loaded plugins: langpacks
Undoing transaction 6, from Sun Feb 16 14:09:51 2014
Install    dapl-2.0.39-2.el7.x86_64          @rhel-7-server-htb-rpms
Dep-Install graphviz-2.30.1-18.el7.x86_64    @rhel-7-server-htb-rpms
Dep-Install graphviz-tcl-2.30.1-18.el7.x86_64 @rhel-7-server-htb-rpms
Install    ibacm-1.0.8-4.el7.x86_64          @rhel-7-server-htb-rpms
Install    ibutils-1.5.7-9.el7.x86_64         @rhel-7-server-htb-rpms
```

```
Dep-Install ibutils-libs-1.5.7-9.el7.x86_64      @rhel-7-server-htb-rpms
...

```

Summary of yum commands

Packages can be located, installed, updated, and removed by name or by package groups.

Task:	Command:
List installed and available packages by name	yum list [NAME-PATTERN]
List installed and available groups	yum grouplist
Search for a package by keyword	yum search KEYWORD
Show details of a package	yum info PACKAGE NAME
Install a package	yum install PACKAGE NAME
Install a package group	yum groupinstall "GROUPNAME"
Update all packages	yum update
Remove a package	yum remove PACKAGE NAME
Display transaction history	yum history

R References

yum(1) and **yum.conf(5)** man pages

Additional information on **yum** may be available in the *Red Hat Enterprise Linux System Administrator's Guide* for Red Hat Enterprise Linux 7, which can be found at
<http://docs.redhat.com/>

Practice: Installing and Updating Software with yum

Guided exercise

In this lab, you will install and remove packages and package groups.

Outcomes:

Explore installing and removing packages with dependencies.

Before you begin...

Reset your serverX system.

1. Search for a specific package.

- 1.1. Attempt to run the command **gnuplot**. You should find that it is not installed.

```
[root@serverX ~]# gnuplot  
bash: gnuplot: command not found...
```

- 1.2. Search for plotting packages.

```
[root@serverX ~]# yum search plot  
Loaded plugins: langpacks  
===== N/S matched: plot =====  
emacs-gnuplot.noarch : Emacs bindings for the gnuplot main application  
gnuplot.x86_64 : A program for plotting mathematical expressions and data  
gnuplot-common.x86_64 : The common gnuplot parts  
python-matplotlib.x86_64 : Python 2D plotting library  
texlive-pst-plot.noarch : Plot data using PSTricks  
  
Name and summary matches only, use "search all" for everything.
```

- 1.3. Find out more information about the **gnuplot** package.

```
[root@serverX ~]# yum info gnuplot  
Name        : gnuplot  
Arch       : x86_64  
...  
...
```

2. Install the **gnuplot** package.

```
[root@serverX ~]# yum install -y gnuplot  
...  
Dependencies Resolved  
  
=====  
 Package      Arch      Version       Repository      Size  
=====  
Installing:  
 gnuplot      x86_64    4.6.2-3.el7   rhel_dvd      645 k  
Installing for dependencies:  
 gnuplot-common x86_64    4.6.2-3.el7   rhel_dvd      595 k
```

```
Transaction Summary
=====
Install 1 Package (+1 Dependent package)
...
```

- 3. Remove packages.
 - 3.1. Attempt to remove the **gnuplot** package, but say no. How many packages would be removed?

```
[root@serverX ~]# yum remove gnuplot
...
Removing:
gnuplot           x86_64     4.6.2-3.el7      @rhel_dvd      1.5 M

Transaction Summary
=====
Remove 1 Package

Installed size: 1.5 M
Is this ok [y/N]: n
```

- 3.2. Attempt to remove the **gnuplot-common** package, but say no. How many packages would be removed?

```
[root@serverX ~]# yum remove gnuplot-common
...
Removing:
gnuplot-common    x86_64     4.6.2-3.el7      @rhel_dvd      1.4 M
Removing for dependencies:
gnuplot          x86_64     4.6.2-3.el7      @rhel_dvd      1.5 M

Transaction Summary
=====
Remove 1 Package (+1 Dependent package)

Installed size: 2.9 M
Is this ok [y/N]: n
```

- 4. Gather information about the "Compatibility Libraries" component group and install it on serverX.
 - 4.1. List all available component groups.

```
[root@serverX ~]# yum grouplist
```

- 4.2. Find out more information about the *Compatibility Libraries* component group, including a list of included packages.

```
[root@serverX ~]# yum groupinfo "Compatibility Libraries"
Loaded plugins: langpacks

Group: Compatibility Libraries
Group-Id: compat-libraries
```

```
Description: Compatibility libraries for applications built on previous
versions of Red Hat Enterprise Linux.
Mandatory Packages:
+compat-db47
+compat-glibc
+compat-libcap1
+compat-libcfc-34
+compat-libgfortran-41
+compat-libtiff3
+compat-openldap
+libpng12
+openssl098e
```

□ 4.3. Install the *Compatibility Libraries* component group.

```
[root@serverX ~]# yum groupinstall "Compatibility Libraries"
Loaded plugins: langpacks
Resolving Dependencies
--> Running transaction check
--> Package compat-db47.x86_64 0:4.7.25-27.el7 will be installed
--> Processing Dependency: compat-db-headers = 4.7.25-27.el7 for package:
compat-db47-4.7.25-27.el7.x86_64
...
Dependencies Resolved

=====
Package           Arch      Version       Repository
=====
Installing for group install "Compatibility Libraries":
compat-db47      x86_64   4.7.25-27.el7    rhel_dvd
libpng12         x86_64   1.2.50-6.el7    rhel_dvd
...
Installing for dependencies:
compat-db-headers      noarch   4.7.25-27.el7    rhel_dvd
...
Transaction Summary
=====
Install  9 Packages (+3 Dependent packages)

Total download size: 5.5 M
Installed size: 21 M
Is this ok [y/d/N]: y
...
Installed:
  compat-db47.x86_64 0:4.7.25-27.el7
  compat-glibc.x86_64 1:2.12-4.el7
...
Dependency Installed:
  compat-db-headers.noarch 0:4.7.25-27.el7

  compat-glibc-headers.x86_64 1:2.12-4.el7

Complete!
```

□ 5. Explore the history and undo options of **yum**.

□ 5.1. Display recent **yum** history.

```
[root@serverX ~]# yum history
```

Loaded plugins: langpacks				
ID	Login user	Date and time	Action(s)	Altered
3	root <root>	2014-06-05 09:33	Install	12
2	root <root>	2014-06-05 09:30	Install	2
1	System <unset>	2014-06-02 20:27	Install	1112 EE

- 5.2. Confirm that the last transaction is the group installation.

```
[root@serverX ~]# yum history info 3
Loaded plugins: langpacks
Transaction ID : 3
Begin time      : Thu Jun  5 09:33:19 2014
Begin rpmdb     : 1210:7c6b529424621773d5fe147315a53d558f726814
End time        :          09:33:40 2014 (21 seconds)
End rpmdb       : 1222:c283bc776b18b9578b87cdec68853f49b31ca0cc
User           : root <root>
Return-Code     : Success
Command Line    : groupinstall Compatibility Libraries
Transaction performed with:
  Installed      rpm-4.11.1-16.el7.x86_64 installed
  Installed      yum-3.4.3-117.el7.noarch installed
Packages Altered:
  Dep-Install    compat-db-headers-4.7.25-27.el7.noarch      @rhel_dvd
  Install        compat-db47-4.7.25-27.el7.x86_64            @rhel_dvd
...
history info
```

- 5.3. Use undo options to remove the last set of packages installed.

```
[root@serverX ~]# yum history undo 3
```

Enabling yum Software Repositories

Objectives

After completing this section, students should be able to enable and disable the use of Red Hat or third-party yum repositories.

Enabling Red Hat software repositories

Registering a system to the subscription management service automatically configures access to software repositories based on the attached subscriptions. To view all available repositories:

```
[root@serverX ~]# yum repolist all
Loaded plugins: langpacks
repo id          repo name           status
rhel-7-public-beta-debug-rpms  Red Hat Enterprise Linux 7 Public  disabled
rhel-7-public-beta-rpms        Red Hat Enterprise Linux 7 Public  enabled: 8,520
rhel-7-public-beta-source-rpms Red Hat Enterprise Linux 7 Public  disabled
repolist: 8,520
```

Enable and disable repositories with **yum-config-manager**. This will change the **enabled** parameter in the **/etc/yum.repos.d/redhat.repo** file.

```
[root@serverX ~]# yum-config-manager --enable rhel-7-public-beta-debug-rpms
Loaded plugins: langpacks
=====
[repo: rhel-7-public-beta-debug-rpms]
=====
async = True
bandwidth = 0
base_persistdir = /var/lib/yum/repos/x86_64/7Server
baseurl = https://cdn.redhat.com/content/beta/rhel/everything/7/x86_64/debug
cache = 0
cachedir = /var/cache/yum/x86_64/7Server/rhel-7-public-beta-debug-rpms
cost = 1000
deltarpm_percentage =
enabled = 1
...
```

Enabling third-party software repositories

Third-party repositories are directories of software package files provided by a non-Red Hat source, which can be accessed by **yum** from a website, FTP server, or local file system. Yum repositories are used by non-Red Hat distributors of software, or for small collections of local packages. (For example, Adobe provides some of its free software for Linux through a yum repository.) The **content.example.com** classroom server actually hosts yum repositories for this class.

Put a file in the **/etc/yum.repos.d/** directory to enable support for a new third-party repository. Repository configuration files must end in **.repo**. The repository definition contains the URL of the repository, a name, whether to use GPG to check the package signatures, and if so, the URL pointing to the trusted GPG key.

Using **yum-config-manager**

If the URL for a yum repository is known, a configuration file can be created with **yum-config-manager**.

```
[root@serverX ~]# yum-config-manager --add-repo="http://dl.fedoraproject.org/pub/epel/beta/7/x86_64/"
Loaded plugins: langpacks
adding repo from: http://dl.fedoraproject.org/pub/epel/beta/7/x86_64/
[dl.fedoraproject.org_pub_epel_beta_7_x86_64_]
name=added from: http://dl.fedoraproject.org/pub/epel/beta/7/x86_64/
baseurl=http://dl.fedoraproject.org/pub/epel/beta/7/x86_64/
enabled=1
```

A file was created in the **/etc/yum.repos.d** directory with the output shown. This file can now be modified to provide a customized name and the location of the GPG key. Administrators should download the key to a local file rather than allowing **yum** to retrieve the key from an external source.

```
[EPEL]
name=EPEL 7 Beta
baseurl=http://dl.fedoraproject.org/pub/epel/beta/7/x86_64/
enabled=1
gpgcheck=1
gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-EPEL-7
```

RPM configuration package for the repository

Some repositories provide this configuration file and GPG public key as part of an RPM package that can be downloaded and installed using **yum localinstall**. One example of this is the volunteer project EPEL (Extra Packages for Enterprise Linux), which provides software not supported by Red Hat but compatible with Red Hat Enterprise Linux.

Installing the Red Hat Enterprise Linux 7 EPEL repo package:

```
[root@serverX ~]# rpm --import http://dl.fedoraproject.org/pub/epel/RPM-GPG-KEY-EPEL-7
[root@serverX ~]# yum install http://dl.fedoraproject.org/pub/epel/beta/7/x86_64/epel-release-7-0.1.noarch.rpm
```

Configuration files often list multiple repository references in a single file. Each repository reference begins with a single-word name in square brackets.

```
[root@serverX ~]# cat /etc/yum.repos.d/epel.repo
[epel]
name=Extra Packages for Enterprise Linux 7 - $basearch
#baseurl=http://download.fedoraproject.org/pub/epel/7/$basearch
mirrorlist=https://mirrors.fedoraproject.org/metalink?repo=epel-7&arch=$basearch
failovermethod=priority
enabled=1
gpgcheck=0
gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-EPEL-7

[epel-debuginfo]
name=Extra Packages for Enterprise Linux 7 - $basearch - Debug
#baseurl=http://download.fedoraproject.org/pub/epel/7/$basearch/debug
mirrorlist=https://mirrors.fedoraproject.org/metalink?repo=epel-debug-7&arch=$basearch
failovermethod=priority
enabled=0
gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-EPEL-7
gpgcheck=1

[epel-source]
```

```
name=Extra Packages for Enterprise Linux 7 - $basearch - Source
#baseurl=http://download.fedoraproject.org/pub/epel/7/SRPMS
mirrorlist=https://mirrors.fedoraproject.org/metalink?repo=epel-source-7&arch=$basearch
failovermethod=priority
enabled=0
gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-EPEL-7
gpgcheck=1
```

The **enabled=0** parameter can be included so that a repository is defined but not searched by default. Repositories can be enabled and disabled persistently with **yum-config-manager** or temporarily with **--enablerepo=PATTERN** and **--disablerepo=PATTERN** options on **yum**.



Warning

Install the RPM GPG key before installing signed packages. This will verify that the packages belong to a key which has been imported. Otherwise, **yum** will complain about the missing key. (The **--nogpgcheck** option can be used to ignore missing GPG keys, but this could cause forged or insecure packages to be installed on the system, potentially compromising its security.)



References

Additional information may be available in the section on configuring yum and yum repositories in the *Red Hat Enterprise Linux System Administrator's Guide* for Red Hat Enterprise Linux 7, which can be found at

<http://docs.redhat.com/>

yum(1), **yum.conf(5)**, and **yum-config-manager(1)** man pages

Practice: Enabling Software Repositories

Guided exercise

In this lab, you will configure your server to use a separate **yum** repository to obtain updates, and update your machine.

Outcomes:

The system will be configured to obtain software updates from a classroom server and the system will be running the latest Linux kernel.

Before you begin...

Reset your serverX system.

1. Configure the system to obtain software from two classroom repositories:

- Classroom packages provided at http://content.example.com/rhel7.0/x86_64/rht
- Updates provided at http://content.example.com/rhel7.0/x86_64/errata

- 1.1. Use **yum-config-manager** to add the classroom packages repository.

```
[root@serverX ~]# yum-config-manager --add-repo="http://
content.example.com/rhel7.0/x86_64/rht"
Loaded plugins: langpacks
adding repo from: http://content.example.com/rhel7.0/x86_64/rht

[content.example.com_rhel7.0_x86_64_rht]
name=added from: http://content.example.com/rhel7.0/x86_64/rht
baseurl=http://content.example.com/rhel7.0/x86_64/rht
enabled=1
```

- 1.2. Create the file **/etc/yum.repos.d/errata.repo** to enable the “Updates” repository with the following content:

```
[updates]
name=Red Hat Updates
baseurl = http://content.example.com/rhel7.0/x86_64/errata
enabled=1
gpgcheck=0
```

2. Use **yum-config-manager** to disable the classroom packages repository.

```
[root@serverX ~]# yum-config-manager --disable
content.example.com_rhel7.0_x86_64_rht
Loaded plugins: langpacks
=====
repo: content.example.com_rhel7.0_x86_64_rht =====
[content.example.com_rhel7.0_x86_64_rht]
...
enabled = 0
...
```

3. Update all relevant software provided using **yum update**.

```
[root@serverX ~]# yum update -y
```

- 4. Verify that there are two versions of the kernel installed. Which version is currently in use?

```
[root@serverX ~]# yum list kernel
```

```
[root@serverX ~]# uname -r
```

- 5. Reboot your serverX, then repeat the previous step. Which version is currently in use?

```
[root@serverX ~]# yum list kernel
```

```
[root@serverX ~]# uname -r
```

- 6. List, then install, the **rht-system** package.

```
[root@serverX ~]# yum list rht*
Loaded plugins: langpacks
Available Packages
rht-system.noarch                      1.0.0-2.el7                  updates
[root@serverX ~]# yum -y install rht-system
Loaded plugins: langpacks
Resolving Dependencies
...
...
```

Examining RPM Package Files

Objectives

After completing this section, students should be able to examine and install downloaded package files.

Examining downloaded packages with rpm

The **rpm** utility is a low-level tool that can get information about the contents of package files and installed packages. It gets its information from a local database or the package files themselves.

The general form of a query is:

- **rpm -q [select-options] [query-options]**
- **rpm --query [select-options] [query-options]**

RPM queries: Select options

- **-q -a**: all installed packages
- **-q *PACKAGENAME***: currently installed *PACKAGENAME*

```
[root@serverX ~]# rpm -q yum
yum-3.4.3-118.el7.noarch
```

- **-q -p *PACKAGEFILE.rpm***: package file named *PACKAGEFILE.rpm*

```
[root@serverX ~]# rpm -q -p http://content.example.com/rhel7.0/x86_64/dvd/Packages/
yum-utils-1.1.31-24.el7.noarch.rpm
yum-utils-1.1.31-24.el7.noarch.rpm
```

- **-q -f *FILENAME***: what package provides *FILENAME*

```
[root@serverX ~]# rpm -q -f /etc/yum/repos.d
yum-3.4.3-118.el7.noarch
```

RPM queries: Information about content of packages

- **-q**: lists the package's name and version; compare to **yum list**
- **-q -i**: package information; compare to **yum info**
- **-q -l**: list of files installed by the specified package

```
[root@serverX ~]# rpm -q -l yum-rhn-plugin
/etc/yum/pluginconf.d/rhnplugin.conf
/usr/share/doc/yum-rhn-plugin-2.0.1
/usr/share/doc/yum-rhn-plugin-2.0.1/LICENSE
/usr/share/locale/zh/LC_MESSAGES/yum-rhn-plugin.mo
...
```

- **-q -c**: list just the configuration files

```
[root@serverX ~]# rpm -q -c yum-rhn-plugin  
/etc/yum/pluginconf.d/rhnplugin.conf
```

- **-q -d**: list just the documentation files

```
[root@serverX ~]# rpm -q -d yum-rhn-plugin  
/usr/share/doc/yum-rhn-plugin-2.0.1/LICENSE  
/usr/share/man/man5/rhnplugin.conf.5.gz  
/usr/share/man/man8/rhnplugin.8.gz  
/usr/share/man/man8/yum-rhn-plugin.8.gz
```

- **-q --scripts**: list shell scripts that may run before or after the package is installed or removed

```
[root@serverX ~]# rpm -q --scripts openssh-server  
preinstall scriptlet (using /bin/sh):  
getent group sshd >/dev/null || groupadd -g 74 -r sshd || :  
getent passwd sshd >/dev/null || \  
    useradd -c "Privilege-separated SSH" -u 74 -g sshd \  
    -s /sbin/nologin -r -d /var/empty/sshd sshd 2> /dev/null || :  
postinstall scriptlet (using /bin/sh):  
  
if [ $1 -eq 1 ] ; then  
    # Initial installation  
    /usr/bin/systemctl preset sshd.service sshd.socket >/dev/null 2>&1 || :  
fi  
preuninstall scriptlet (using /bin/sh):  
  
if [ $1 -eq 0 ] ; then  
    # Package removal, not upgrade  
    /usr/bin/systemctl --no-reload disable sshd.service sshd.socket > /dev/null  
    2>&1 || :  
    /usr/bin/systemctl stop sshd.service sshd.socket > /dev/null 2>&1 || :  
fi  
postuninstall scriptlet (using /bin/sh):  
  
/usr/bin/systemctl daemon-reload >/dev/null 2>&1 || :  
if [ $1 -ge 1 ] ; then  
    # Package upgrade, not uninstall  
    /usr/bin/systemctl try-restart sshd.service >/dev/null 2>&1 || :  
fi
```

- **-q --changelog**: list change information for the package

```
[root@serverX ~]# rpm -q --changelog audit  
* Thu Oct 03 2013 Steve Grubb <sgrubb@redhat.com> 2.3.2-3  
resolves: #828495 - semanage port should generate an audit event  
  
* Thu Aug 29 2013 Steve Grubb <sgrubb@redhat.com> 2.3.2-2  
resolves: #991056 - ausearch ignores USER events with -ua option  
...
```

Querying and installing package files:

```
[root@serverX ~]# wget http://classroom/pub/materials/wonderwidgets-1.0-4.x86_64.rpm
```

```
[root@serverX ~]# rpm -q -l -p wonderwidgets-1.0-4.x86_64.rpm
/etc/wonderwidgets.conf
/usr/bin/wonderwidgets
/usr/share/doc/wonderwidgets-1.0
/usr/share/doc/wonderwidgets-1.0/README.txt
```



Note

The **repoquery** command can also be used to get information about packages and their contents. It differs from **rpm** by looking up that information in yum's repositories instead of the local database of installed packages.

Using yum to install local package files

yum localinstall PACKAGEFILE.rpm can be used to install package files directly. It automatically downloads any dependencies the package has from any configured **yum** repositories. Packages are normally digitally signed to ensure they are legitimate; if the package is not signed by a key trusted by your system, it will be rejected. The **--nogpgcheck** option can disable the signature check if you are certain the package is legitimate if a package is installed with **yum install**. In Red Hat Enterprise Linux 7 the **--nogpgcheck** option is not required if a package is installed with **yum localinstall**.

```
[root@serverX ~]# yum localinstall wonderwidgets-1.0-4.x86_64.rpm
[root@serverX ~]# rpm -q wonderwidgets
wonderwidgets-1.0-4.x86_64
```



Note

rpm -ivh PACKAGEFILE.rpm can also be used to install package files. However, using **yum** helps maintain a transaction history kept by **yum** (see **yum history**).



Warning

Be careful when installing packages from third parties, not just because of the software that they may install, but because the RPM may run arbitrary scripts as **root** as part of the installation process.

Extracting files from RPM packages

Files in the RPM package can be extracted without installing the package using **cpio**, an archiving tool like **zip** or **tar**. Pipe the output of **rpm2cpio PACKAGEFILE.rpm** into **cpio -id**, and it will extract all the files stored in the RPM package. Subdirectory trees will be created as needed, relative to the current working directory.

Select files can also be extracted by specifying the path of the file:

```
[root@serverX ~]# rpm2cpio wonderwidgets-1.0-4.x86_64.rpm | cpio -id "*txt"
11 blocks
[root@serverX ~]# ls -l usr/share/doc/wonderwidgets-1.0/
```

```
total 4
-rw-r--r--. 1 root root 76 Feb 13 19:27 README.txt
```

Summary of rpm query commands

Installed packages can be queried directly with **rpm** command. Add a **-p** option to query a package file before installation.

Task:	Command:
Display information about a package	rpm -q -i NAME
List all files included in a package	rpm -q -l NAME
List configuration files included in a package	rpm -q -c NAME
List documentation files included in a package	rpm -q -d NAME
Show a short summary of the reason for a new package release	rpm -q --changelog NAME
Display the shell scripts included in a package	rpm -q --scripts NAME



References

yum(8), **rpm(8)**, **repoquery(1)**, **rpm2cpio(8)**, and **cpio(1)** man pages

Practice: Working with RPM Package Files

Guided exercise

In this lab, you will gather information about a third-party package, extract files from it, and install it on your serverX system.

Outcomes:

A package not provided by a yum repository is installed on the system.

Before you begin...

Reset your serverX system.

1. Download *wonderwidgets-1.0-4.x86_64.rpm* from <http://classroom/pub/materials>.

```
[root@serverX ~]# wget http://classroom/pub/materials/
wonderwidgets-1.0-4.x86_64.rpm
--2014-02-11 15:58:02-- http://classroom/pub/materials/
wonderwidgets-1.0-4.x86_64.rpm
Resolving classroom (classroom)... 172.25.0.254
Connecting to classroom (classroom)|172.25.0.254|:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 5000 (4.9K) [application/x-rpm]
Saving to: 'wonderwidgets-1.0-4.x86_64.rpm'

100%[=====] 5,000      --.-K/s   in 0s

2014-02-11 15:58:02 (381 MB/s) - 'wonderwidgets-1.0-4.x86_64.rpm' saved
[5000/5000]
```

2. What files does it contain?

```
[root@serverX ~]# rpm -q -p wonderwidgets-1.0-4.x86_64.rpm -l
/etc/wonderwidgets.conf
/usr/bin/wonderwidgets
/usr/share/doc/wonderwidgets-1.0
/usr/share/doc/wonderwidgets-1.0/README.txt
```

3. What scripts does it contain?

```
[root@serverX ~]# rpm -q -p wonderwidgets-1.0-4.x86_64.rpm --scripts
```

4. How much disk space will it use when installed?

```
[root@serverX ~]# rpm -q -p wonderwidgets-1.0-4.x86_64.rpm -i
Name        : wonderwidgets
Version     : 1.0
Release     : 4
Architecture: x86_64
Install Date: (not installed)
Group       : GLS/Applications
Size : 4849
License     : GPL
```

```
Signature : (none)
Source RPM : wonderwidgets-1.0-4.src.rpm
Build Date : Fri 03 Dec 2010 05:42:55 AM EST
Build Host : station166.rosemont.lan
Relocations : (not relocatable)
Vendor : Red Hat, Inc.
Summary : Demonstration package for use in GLS training.
Description :
A demonstration package that installs an executable, and a config file.
```

- 5. Use **yum localinstall** to install the package.

```
[root@serverX ~]# yum localinstall wonderwidgets-1.0-4.x86_64.rpm
Loaded plugins: langpacks
Examining wonderwidgets-1.0-4.x86_64.rpm: wonderwidgets-1.0-4.x86_64
Marking wonderwidgets-1.0-4.x86_64.rpm to be installed
Resolving Dependencies
--> Running transaction check
---> Package wonderwidgets.x86_64 0:1.0-4 will be installed
---> Finished Dependency Resolution

Dependencies Resolved

=====
Package      Arch      Version       Repository      Size
=====
Installing:
wonderwidgets   x86_64    1.0-4        /wonderwidgets-1.0-4.x86_64  4.7 k

Transaction Summary
=====
Install 1 Package

Total size: 4.7 k
Installed size: 4.7 k
Is this ok [y/d/N]: y
Downloading packages:
Running transaction check
Running transaction test
Transaction test succeeded
Running transaction
  Installing : wonderwidgets-1.0-4.x86_64                               1/1
  Verifying  : wonderwidgets-1.0-4.x86_64                               1/1

Installed:
  wonderwidgets.x86_64 0:1.0-4

Complete!
```

Lab: Installing and Updating Software Packages

Performance checklist

In this lab, you will install and update select software packages.

Outcomes:

New and updated packages are installed on the system.

Before you begin...

Reset your serverX system.

1. Create the file **/etc/yum.repos.d/errata.repo**, to enable the "Updates" repository found on the content machine. It should access content found at the following URL: http://content.example.com/rhel7.0/x86_64/errata. Do *not* check GPG signatures.
2. Configure serverX to adhere to very specific software requirements. It must have the latest version of the following packages installed. Do not install all updates. Only install updates for the packages listed if they are available.
 - 2.1. **kernel** (existing package w/ an update)
 - 2.2. **xsane-gimp** (new package)
 - 2.3. **rht-system** (new package)
3. For security reasons, it should not have the **wvdial** package installed.
4. When you are ready to check your work, run **lab software grade** on serverX.

Solution

In this lab, you will install and update select software packages.

Outcomes:

New and updated packages are installed on the system.

Before you begin...

Reset your serverX system.

1. Create the file **/etc/yum.repos.d/errata.repo**, to enable the “Updates” repository found on the content machine. It should access content found at the following URL: http://content.example.com/rhel7.0/x86_64/errata. Do *not* check GPG signatures.

Create the file **/etc/yum.repos.d/errata.repo** with the following content:

```
[updates]
name=Red Hat Updates
baseurl=http://content.example.com/rhel7.0/x86_64/errata
enabled=1
gpgcheck=0
```

2. Configure serverX to adhere to very specific software requirements. It must have the latest version of the following packages installed. Do not install all updates. Only install updates for the packages listed if they are available.
 - 2.1. **kernel** (existing package w/ an update)

```
yum update kernel
```

- 2.2. **xsane-gimp** (new package)

```
yum install xsane-gimp
```

- 2.3. **rht-system** (new package)

```
yum install rht-system
```

3. For security reasons, it should not have the **wvdial** package installed.

```
yum remove wvdial
```

4. When you are ready to check your work, run **lab software grade** on serverX.

```
[student@serverX ~]$ lab software grade
```

Summary

Attaching Systems to Subscriptions for Software Updates

Registering systems allows access to software updates for installed products.

RPM Software Packages and Yum

RPM packages, grouped into yum repositories, provide a uniform method of tracking software installation and updates.

Managing Software Updates with **yum**

yum is used to install and update software packages.

Enabling **yum** Software Repositories

Repositories for **yum** are configured in the **/etc/yum.repos.d** directory.

Examining RPM Package Files

Packages downloaded outside of yum repositories can be queried and installed with **rpm**.



CHAPTER 14

ACCESSING LINUX FILE SYSTEMS

Overview	
Goal	To access and inspect existing file systems on a Red Hat Enterprise Linux system.
Objectives	<ul style="list-style-type: none">Identify the file system hierarchy.Access the contents of file systems.Use hard links and symlinks to make multiple names.Search for files on mounted file systems.
Sections	<ul style="list-style-type: none">Identifying File Systems and Devices (and Practice)Mounting and Unmounting File Systems (and Practice)Making Links Between Files (and Practice)Locating Files on the System (and Practice)
Lab	<ul style="list-style-type: none">Accessing Linux File Systems

Identifying File Systems and Devices

Objectives

After completing this section, students should be able to identify a directory in the file system hierarchy and what storage device it is stored on.

Storage management concepts

A file system is an organized structure of data-holding files and directories residing on a storage device, such as a physical disk or partition. The file system hierarchy discussed earlier assembles all the file systems into one tree of directories with a single root, the `/` directory. The advantage here is that the existing hierarchy can be extended at any time by adding a new disk or partition containing a supported file system to add disk space anywhere in the file system tree. The process of adding a new file system to the existing directory tree is called *mounting*. The directory where the new file system is mounted is referred to as a *mount point*. This is a fundamentally different concept than that used on a Microsoft Windows system, where a new file system is represented by a separate drive letter.

Hard disks and storage devices are normally divided up into smaller chunks called *partitions*. A partition is a way to compartmentalize a disk. Different parts of it can be formatted with different file systems or used for different purposes. For example, one partition could contain user home directories while another could contain system data and logs. If a user fills up the home directory partition with data, the system partition may still have space available. Placing data in two separate file systems on two separate partitions helps in planning data storage.

Storage devices are represented by a special file type called *block device*. The block device is stored in the `/dev` directory. In Red Hat Enterprise Linux, the first SCSI, PATA/SATA, or USB hard drive detected is `/dev/sda`, the second is `/dev/sdb`, and so on. This name represents the whole drive. The first primary partition on `/dev/sda` is `/dev/sda1`, the second partition is `/dev/sda2`, and so on.

A long listing of the `/dev/vda` device file on serverX reveals its special file type as **b**, which stands for block device:

```
[student@serverX ~]$ ls -l /dev/vda
brw-rw----. 1 root disk 253, 0 Mar 13 08:00 /dev/vda
```

Note

Exceptions are hard drives in virtual machines, which typically show up as `/dev/vd<letter>` or `/dev/xvd<letter>`.

Another way of organizing disks and partitions is with *logical volume management* (LVM). With LVM, one or more block devices can be aggregated into a storage pool called a *volume group*. Disk space is made available with one or more *logical volumes*. A logical volume is the equivalent of a partition residing on a physical disk. Both the volume group and the logical volume have names assigned upon creation. For the volume group, a directory with the same name as the volume group exists in the `/dev` directory. Below that directory, a symbolic link with the same name as the logical volume has been created. For example, the device file representing the `mylv` logical volume in the `myvg` volume group is `/dev/myvg/mylv`.

It should be noted that LVM relies on the *Device Mapper* (DM) kernel driver. The above symbolic link **/dev/myvg/mylv** points to the **/dev/dm-number** block device node. The assignment of the *number* is sequential beginning with zero (0). There is another symbolic link for every logical volume in the **/dev/mapper** directory with the name **/dev/mapper/myvg-my lv**. Access to the logical volume can generally use either of the consistent and reliable symbolic link names, since the **/dev/dm-number** name can vary with each boot.

Examining file systems

To get an overview about the file system mount points and the amount of free space available, run the **df** command. When the **df** command is run without arguments, it will report total disk space, used disk space, and free disk space on all mounted regular file systems. It will report on both local and remote systems and the percentage of the total disk space that is being used.

Display the file systems and mount points on the serverX machine.

```
[student@serverX ~]$ df
Filesystem      1K-blocks    Used Available Use% Mounted on
/dev/vda1        6240256  4003760   2236496  65% /
devtmpfs         950536      0   950536   0% /dev
tmpfs            959268     80   959188   1% /dev/shm
tmpfs            959268    2156   957112   1% /run
tmpfs            959268      0   959268   0% /sys/fs/cgroup
```

The partitioning on the serverX machine shows one real file system, which is mounted on **/**. This is common for virtual machines. The **tmpfs** and **devtmpfs** devices are file systems in system memory. All files written into **tmpfs** or **devtmpfs** disappear after system reboot.

To improve readability of the output sizes, there are two different *human-readable* options: **-h** or **-H**. The difference between these two options is that **-h** will report in KiB (2^{10}), MiB (2^{20}), or GiB (2^{30}), while the **-H** option will report in SI units: KB (10^3), MB (10^6), GB (10^9), etc. Hard drive manufacturers usually use SI units when advertising their products.

Show a report on the file systems on the serverX machine with all units converted to human-readable format:

```
[student@serverX ~]$ df -h
Filesystem      Size  Used Avail Use% Mounted on
/dev/vda1       6.0G  3.9G  2.2G  65% /
devtmpfs        929M    0  929M   0% /dev
tmpfs           937M   80K  937M   1% /dev/shm
tmpfs           937M   2.2M  935M   1% /run
tmpfs           937M    0  937M   0% /sys/fs/cgroup
```

For more detailed information about the space used by a certain directory tree, there is the **du** command. The **du** command has **-h** and **-H** options to convert the output to human-readable format. The **du** command shows the size of all files in the current directory tree recursively.

Show a disk usage report for the **/root** directory on serverX:

```
[root@serverX ~]# du /root
4 /root/.ssh
4 /root/.cache/dconf
4 /root/.cache
4 /root/.dbus/session-bus
```

```
4 /root/.dbus
0 /root/.config/ibus/bus
0 /root/.config/ibus
0 /root/.config
14024 /root
```

Show a disk usage report in human-readable format for the **/var/log** directory on serverX:

```
[root@serverX ~]# du -h /var/log
...
4.9M /var/log/sa
68K /var/log/prelink
0 /var/log/qemu-ga
14M /var/log
```

References

df(1) and **du(1)** man pages

Practice: Identifying File Systems and Devices

Quiz

Match the following items to their counterparts in the table.

/dev/sda2	/dev/sdb3	/dev/sdc	/dev/vdb	/dev/vdb3
/dev/vg_install/lv_home				

Description	Device file
The device file of a SATA hard drive residing in /dev .	
The device file of the second partition on the first SATA hard drive in /dev .	
The device file of a logical volume in /dev .	
The device file of the second disk in a virtual machine in /dev .	
The device file of the third partition on the second SATA hard drive in /dev .	
The device file of the third partition on the second disk in a virtual machine in /dev .	

Solution

Match the following items to their counterparts in the table.

Description	Device file
The device file of a SATA hard drive residing in /dev .	/dev/sdc
The device file of the second partition on the first SATA hard drive in /dev .	/dev/sda2
The device file of a logical volume in /dev .	/dev/vg_install/lv_home
The device file of the second disk in a virtual machine in /dev .	/dev/vdb
The device file of the third partition on the second SATA hard drive in /dev .	/dev/sdb3
The device file of the third partition on the second disk in a virtual machine in /dev .	/dev/vdb3

Mounting and Unmounting File Systems

Objectives

After completing this section, students should be able to access the contents of file systems by adding and removing file systems from the file system hierarchy.

Mounting file systems manually

A file system residing on a SATA/PATA or SCSI device needs to be mounted manually to access it. The **mount** command allows the root user to manually mount a file system. The first argument of the **mount** command specifies the file system to mount. The second argument specifies the target directory where the file system is made available after mounting it. The target directory is referred to as a mount point.

The **mount** command expects the file system argument in one of two different ways:

- The device file of the partition holding the file system, residing in **/dev**.
- The *UUID*, a universal unique identifier of the file system.



Note

As long as a file system is not recreated, the UUID stays the same. The device file can change; for example, if the order of the devices is changed or if additional devices are added to the system.

The **blkid** command gives an overview of existing partitions with a file system on them and the UUID of the file system, as well as the file system used to format the partition.

```
[root@serverX ~]# blkid
/dev/vda1: UUID="46f543fd-78c9-4526-a857-244811be2d88" TYPE="xfs"
```



Note

A file system can be mounted on an existing directory. The **/mnt** directory exists by default and provides an entry point for mount points. It is used for manually mounting disks. It is recommended to create a subdirectory under **/mnt** and use that subdirectory as a mount point unless there is a reason to mount the file system in another specific location in the file system hierarchy.

Mount by device file of the partition that holds the file system.

```
[root@serverX ~]# mount /dev/vdb1 /mnt/mydata
```

Mount the file system by universal unique id, or the UUID, of the file system.

```
[root@serverX ~]# mount UUID="46f543fd-78c9-4526-a857-244811be2d88" /mnt/mydata
```



Note

If the directory acting as mount point is not empty, the files that exists in that directory are not accessible as long as a file system is mounted there. All files written to the mount point directory end up on the file system mounted there.

Unmounting file systems

To unmount a file system, the **umount** command expects the mount point as an argument.

Change to the **/mnt/mydata** directory. Try to umount the device mounted on the **/mnt/mydata** mount point. It will fail.

```
[root@serverX ~]# cd /mnt/mydata
[root@serverX mydata]# umount /mnt/mydata
umount: /mnt/mydata: target is busy.
        (In some cases useful info about processes that use
         the device is found by lsof(8) or fuser(1))
```

Unmounting is not possible if the mount point is accessed by a process. For **umount** to be successful, the process needs to stop accessing the mount point.

The **lsof** command lists all open files and the process accessing them in the provided directory. It is useful to identify which processes currently prevent the file system from successful unmounting.

```
[root@serverX mydata]# lsof /mnt/mydata
COMMAND  PID USER   FD   TYPE DEVICE SIZE/OFF NODE NAME
bash    1593 root cwd   DIR  253,2       6  128 /mnt/mydata
lsof    2532 root cwd   DIR  253,2      19  128 /mnt/mydata
lsof    2533 root cwd   DIR  253,2      19  128 /mnt/mydata
```

Once the processes are identified, an action can be taken, such as waiting for the process to complete or sending a SIGTERM or SIGKILL signal to the process. In this case, it is sufficient to change the current working directory to a directory outside the mount point.

```
[root@serverX mydata]# cd
[root@serverX ~]# umount /mnt/mydata
```



Note

A common cause for the file system on the mount point to be busy is if the current working directory of a shell prompt is below the active mount point. The process accessing the mount point is **bash**. Changing to a directory outside the mount point allows the device to be unmounted.

Accessing removable storage devices

Removable media, such as USB flash devices and drives, get automatically mounted by the graphical desktop environment when plugged in. The mount point for the removable medium is

/run/media/<user>/<label>. The <user> is the user logged into the graphical environment. The <label> is the name given to the file system when it was created.



Warning

To safely remove USB media from the system, it is required to unmount it before physically removing it from the USB slot to synchronize the file system. Removing a USB storage device without unmounting the file system on it can result in data loss.



References

mount(8), **umount(8)**, and **lsof(8)** man pages

Practice: Mounting and Unmounting File Systems

Guided exercise

In this lab, you will mount and unmount file systems.

Outcomes:

The user identifies and mounts a new file system at a specified mount point, then unmounts it.

Before you begin...

Reset your serverX system. Run the script **lab fs setup** before starting the exercise.

- 1. A new partition with a file system has been added to the second disk (vdb) on your serverX machine. Mount the newly available partition by UUID at the newly created mount point **/mnt/newspace**.
 - 1.1. Use the **blkid** to discover the UUID of the newly added partition, **vdb1**, on serverX.

```
[root@serverX ~]# blkid  
/dev/vda1: UUID="46f543fd-78c9-4526-a857-244811be2d88" TYPE="xfs"  
/dev/vdb1: UUID="7c5e3fbb-34eb-4431-a4a5-9b887c1b6866" TYPE="xfs"
```

- 1.2. Create the mount point **/mnt/newspace** on serverX.

```
[root@serverX ~]# mkdir /mnt/newspace
```

- 1.3. Mount the file system by UUID on the **/mnt/newspace** directory of the serverX machine.

```
[root@serverX ~]# mount UUID="7c5e3fbb-34eb-4431-a4a5-9b887c1b6866" /mnt/  
newspace
```

- 2. Change to the **/mnt/newspace** directory and create a new directory, **/mnt/newspace/newdir**, with an empty file, **/mnt/newspace/newdir/newfile**, on serverX.

- 2.1. Change to the **/mnt/newspace** directory on serverX.

```
[root@serverX ~]# cd /mnt/newspace
```

- 2.2. Create a new directory, **/mnt/newspace/newdir**, on serverX.

```
[root@serverX newspace]# mkdir newdir
```

- 2.3. Create a new empty file, **/mnt/newspace/newdir/newfile**, on serverX.

```
[root@serverX newspace]# touch newdir/newfile
```

- 3. Unmount the file system mounted on the **/mnt/newspace** directory on serverX.
 - 3.1. Try to umount **/mnt/newspace** while the current directory on the shell is still **/mnt/newspace** on serverX.

```
[root@serverX newspace]# umount /mnt/newspace
```

- 3.2. Change the current directory on the shell to **/root**.

```
[root@serverX newspace]# cd  
[root@serverX ~]#
```

- 3.3. Successfully umount **/mnt/newspace** on serverX.

```
[root@serverX ~]# umount /mnt/newspace
```

Making Links Between Files

Objectives

After completing this section, students should be able to use hard links and soft links to make multiple names point to the same file.

Managing links between files

Creating hard links

A hard link is a new directory entry with a reference to an existing file on the file system. Every file in a file system has one hard link by default. To save space, instead of copying, a new hard link can be created to reference the same file. A new hard link either needs to have a different file name, if it is created in the same directory as the existing hard link, or it needs to reside in a different directory. All hard links pointing to the same file have the same permissions, link count, user/group ownerships, time stamps, and file content. Hard links pointing to the same file content need to be on the same file system.

The **ls -l** shows the hard link count after the permissions and before the owner of a file.

```
[root@serverX ~]# echo "Hello World" > newfile.txt
[root@serverX ~]# ls -l newfile.txt
-rw-r--r--. 1 root root 0 Mar 11 19:19 newfile.txt
```

The command **ln** creates new hard links to existing files. The command expects an existing file as the first argument, followed by one or more additional hard links. The hard links can reside anywhere as long as they are on the same file system as the existing file. After a new hard link is created, there is no way to tell which of the existing hard links is the original one.

Create a hard link **newfile-link2.txt** for the existing file **newfile.txt** in the same directory.

```
[root@serverX ~]# ln newfile.txt /tmp/newfile-hlink2.txt
[root@serverX ~]# ls -l newfile.txt /tmp/newfile-hlink2.txt
-rw-rw-r--. 2 root root 12 Mar 11 19:19 newfile.txt
-rw-rw-r--. 2 root root 12 Mar 11 19:19 newfile-hlink2.txt
```

Even if the original file gets deleted, the content of the file is still available as long as at least one hard link exists.

```
[root@serverX ~]# rm -f newfile.txt
[root@serverX ~]# ls -l /tmp/newfile-link2.txt
-rw-rw-r--. 1 root root 12 Mar 11 19:19 /tmp/newfile-link2.txt
[root@serverX ~]# cat /tmp/newfile-link2.txt
Hello World
```



Important

All hard links referencing the same file have the same permissions, link count, user/group ownerships, time stamps, and file content. If any of that information is changed on one hard link, all other hard links pointing at the same file will show the new information as well.

Creating soft links

The **ln -s** command creates a soft link, also known as a symbolic link. A soft link is a special file type that points to an existing file or directory. Soft links can point to a file or directory on another file system. Unlike a hard link, a symbolic link can point to a file on a different file system.

```
[root@serverX ~]# ln -s /root/newfile-link2.txt /tmp/newfile-symlink.txt
[root@serverX ~]# ls -l newfile-link2.txt /tmp/newfile-symlink.txt
lrwxrwxrwx. 1 root root 11 Mar 11 20:59 /tmp/newfile-symlink.txt -> /root/newfile-link2.txt
-rw-rw-r--. 1 root root 12 Mar 11 19:19 newfile-link2.txt
```

When the original file gets deleted, the soft link is still pointing to the file but the target is gone. A soft link pointing to a missing file is called a "dangling soft link."

```
[root@serverX ~]# rm -f newfile-link2.txt
[root@serverX ~]# ls -l /tmp/newfile-symlink.txt
lrwxrwxrwx. 1 root root 11 Mar 11 20:59 /tmp/newfile-symlink.txt -> newfile-link2.txt
[root@serverX ~]# cat /tmp/newfile-symlink.txt
cat: /tmp/newfile-symlink.txt: No such file or directory
```

A soft link can point to a directory. The soft link then acts like a directory. Changing to the soft link directory with **cd** works as expected.

Create a soft link **/root/configfiles** pointing to the **/etc** directory.

```
[root@serverX ~]# ln -s /etc /root/configfiles
[root@serverX ~]# cd /root/configfiles
[root@serverX configfiles]# pwd
/root/configfiles
```

R

References

ln(1) man page

Practice: Making Links Between Files

Guided exercise

In this lab, you will create hard and soft links.

Outcomes:

The user creates a hard link and a soft link.

- 1. Create an additional hard link **/root/qmp-manual.txt** for the existing file **/usr/share/doc/qemu-kvm/qmp-commands.txt** on serverX.
 - 1.1. Create the hard link **/root/qmp-manual.txt**. Link it to the file **/usr/share/doc/qemu-kvm/qmp-commands.txt**.

```
[root@serverX ~]# ln /usr/share/doc/qemu-kvm/qmp-commands.txt /root/qmp-manual.txt
```

- 1.2. Verify the link count on the newly created link **/root/qmp-manual.txt**.

```
[root@serverX ~]# ls -l /root/qmp-manual.txt  
-rw-r--r--. 2 root root 63889 Nov 11 02:58 /root/qmp-manual.txt
```

- 1.3. Verify the link count on the original file **/usr/share/doc/qemu-kvm/qmp-commands.txt**.

```
[root@serverX ~]# ls -l /usr/share/doc/qemu-kvm/qmp-commands.txt  
-rw-r--r--. 2 root root 63889 Nov 11 02:58 /usr/share/doc/qemu-kvm/qmp-commands.txt
```

- 2. Create the soft link **/root/tempdir** pointing to the directory **/tmp** on serverX.

- 2.1. Create the soft link **/root/tempdir**. Link it to **/tmp**.

```
[root@serverX ~]# ln -s /tmp /root/tempdir
```

- 2.2. Verify the newly created link with **ls -l**.

```
[root@serverX ~]# ls -l /root  
lrwxrwxrwx. 1 root root 4 Mar 13 08:42 /root/tempdir -> /tmp
```

Locating Files on the System

Objectives

After completing this section, students should be able to search for files on mounted file systems using **find** and **locate**.

Tools for finding files

A system administrator needs tools for searching files matching certain criteria on the file system. This section discusses two commands that can search files in the file system. The **locate** command searches a pregenerated database for file names or file paths and returns the results instantly. The **find** command searches the file system in real time by crawling through the file system.

Locating files by name with locate

The **locate** command returns search results based on file name or path from the **locate** database. The database stores file name and path information.

When searching entries as a regular user, results are returned only for where the user invoking the **locate** search has read permissions on the directory tree that contains the matching element.

Search for files with "passwd" in the name or path in directory trees readable by user student on serverX.

```
[student@serverX ~]$ locate passwd
/etc/passwd
/etc/passwd-
/etc/pam.d/passwd
/etc/security/opasswd
/usr/bin/gpasswd
/usr/bin/grub2-mkpasswd-pbkdf2
/usr/bin/lppasswd
/usr/bin/passwd
/usr/bin/userpasswd
/usr/bin/vino-passwd
/usr/bin/vncpasswd
```

Results are returned even when the file name or path is only a partial match to the search query.

```
[root@serverX ~]# locate image
/home/myuser/boot.image
/home/someuser/my_family_image.png
/home/student/myimages-vacation/picture.png
```

The **-i** option performs a case-insensitive search. With this option, all possible combinations of upper- and lowercase letters match the search.

```
[student@serverX ~]$ locate -i messages
...
/usr/share/vim/vim74/lang/zh_TW/LC_MESSAGES
```

```
/usr/share/vim/vim74/lang/zh_TW/LC_MESSAGES/vim.mo  
/usr/share/vim/vim74/lang/zh_TW.UTF-8/LC_MESSAGES  
/usr/share/vim/vim74/lang/zh_TW.UTF-8/LC_MESSAGES/vim.mo  
/usr/share/vim/vim74/syntax/messages.vim  
/usr/share/vim/vim74/syntax/msmessages.vim  
/var/log/messages
```

The **-n** option limits the number of returned search results by **locate**. The following example limits the search results returned by **locate** to the first five matches.

```
[student@serverX ~]$ locate -n 5 snow.png  
/usr/share/icons/HighContrast/16x16/status/weather-snow.png  
/usr/share/icons/HighContrast/22x22/status/weather-snow.png  
/usr/share/icons/HighContrast/24x24/status/weather-snow.png  
/usr/share/icons/HighContrast/256x256/status/weather-snow.png  
/usr/share/icons/HighContrast/32x32/status/weather-snow.png
```

Note

The **locate** database is automatically updated every day. The root user can perform an update of the database with the **updatedb** command.

```
[root@serverX ~]# updatedb
```

Searching for files with find

The **find** command performs a real-time search in the local file systems to find files that match the criteria of the command-line arguments. The **find** command is looking at files in the file system as your user account. The user invoking the **find** command must have read and execute permission on a directory to examine its contents.

The first argument to the **find** command is the directory to search. If the directory argument is omitted, **find** will start the search in the current directory and look for matches in any of the subdirectories.

To search the home directory of user student, give **find** a starting directory of **/home/student**. To search the entire system, provide a starting directory of **/**.

Note

find has a huge number of options that can be provided to describe exactly what kind of file should be found. Searches can be based on file name, file size, last modified time stamp, and other file characteristics in any combination.

The **-name** option followed by a file name looks up files matching the given file name and returns all exact matches. To search for files named **sshd_config** in the **/** directory and all subdirectories on serverX, run:

```
[root@serverX ~]# find / -name sshd_config  
/etc/ssh/sshd_config
```

Wild cards are available to search for a file name and return all results that are a partial match. When using wild cards, it is important to quote the file name to look for to prevent the terminal from interpreting the wild card.

The following example searches for files in the / directory on serverX that end in .txt:

```
[root@serverX ~]# find / -name '*.txt'  
/etc/pki/nssdb/pkcs11.txt  
/etc/brltty/brl-lt-all.txt  
/etc/brltty/brl-mb-all.txt  
/etc/brltty/brl-md-all.txt  
/etc/brltty/brl-mn-all.txt  
...
```

To search for files in /etc/ that contain **pass** anywhere in their names on serverX, run:

```
[root@serverX ~]# find /etc -name '*pass*'  
/etc/passwd  
/etc/passwd-  
/etc/fonts/conf.d/60-overpass.conf  
/etc/selinux/targeted/modules/active/modules/passenger.pp  
/etc/security/opasswd  
/etc/pam.d/passwd  
/etc/pam.d/password-auth-ac  
/etc/pam.d/password-auth  
/etc/pam.d/gdm-password
```

To perform a case-insensitive search for a given file name, use the **-iname** option, followed by the file name to search. To search case-insensitively for files that have **messages** in their names in the / directory on serverX, run:

```
[root@serverX ~]# find / -iname '*messages*'  
/var/log/messages  
/usr/lib64/python2.7/site-packages/orca/notification_messages.py  
/usr/lib64/python2.7/site-packages/orca/notification_messages.pyc  
/usr/lib64/python2.7/site-packages/orca/notification_messages.pyo  
/usr/share/locale/aa/LC_MESSAGES  
/usr/share/locale/ab/LC_MESSAGES  
/usr/share/locale/ace/LC_MESSAGES
```

find can search for files based on their ownership or permissions. Useful options when searching by owner are **-user** and **-group**, which search by name, and **-uid** and **-gid**, which search by ID.

Search for files owned by the user **student** in the /home/**student** directory on serverX.

```
[student@serverX ~]$ find -user student  
.brash_logout  
.brash_profile  
.brashrc  
.ssh  
...
```

Search for files owned by the group **student** in /home/ in the /home/**student** directory on serverX.

Chapter 14. Accessing Linux File Systems

```
[student@serverX ~]$ find -group student
.
./.bash_logout
./.bash_profile
./.bashrc
./.ssh
...
```

Search for files owned by user ID 1000 in the **/home/student** directory on serverX.

```
[student@serverX ~]$ find -uid 1000
.
./.bash_logout
./.bash_profile
./.bashrc
./.ssh
...
```

Search for files owned by group ID 1000 in the **/home/student** directory on serverX.

```
[student@serverX ~]$ find -gid 1000
.
./.bash_logout
./.bash_profile
./.bashrc
./.ssh
...
```

Search for files owned by user *root* and group *mail* on the serverX machine.

```
[root@serverX ~]# find / -user root -group mail
/var/spool/mail
/var/spool/mail/root
```

The **-perm** option is used to look for files with a particular set of permissions. Permissions can be described as octal values, with some combination of 4, 2, and 1 for read, write, and execute. Permissions can be preceded by a / or - sign.

A numeric permission preceded by / will match files that have at least one bit of user, group, or other for that permission set. A file with permissions **r--r--r--** does not match /222, but one with **rw-r--r--** does. A - sign before a permission means that all three instances of that bit must be on, so neither of the previous examples would match, but something like **rw-rw-rw-** would.

To use a more complex example, the following command would match any file for which the user has read, write, and execute permissions, members of the group have read and write permissions, and others have read-only access:

```
[root@serverX ~]# find /home -perm 764
```

To match files for which the user has at least write and execute permissions, *and* the group has at least write permissions, *and* others have at least read access:

```
[root@serverX ~]# find /home -perm -324
```

To match files for which the user has read permissions, or the group has at least read permissions, or others have at least write access:

```
[root@serverX ~]# find /home -perm /442
```

When used with / or -, a value of 0 works like a wild card, since it means "a permission of at least nothing."

To match any file in the **/home/student** directory for which others have at least read access on serverX, run:

```
[student@serverX ~]$ find -perm -004
```

Find all files in the **/home/student** directory where *other* has write permissions on serverX.

```
[student@serverX ~]$ find -perm -002
```

The **find** command can look up files that match a size specified with the **-size** option, followed by a numerical value and the unit.

Units to be used with the **-size** option are:

- k, for kilobyte
- M, for megabyte
- G, for gigabyte

Search for files with a size of *exactly* 10 megabytes.

```
[student@serverX ~]$ find -size 10M
```

Find files with a size *more than* 10 gigabytes.

```
[student@serverX ~]$ find -size +10G
```

List all files with a size *less than* 10 kilobytes.

```
[student@serverX ~]$ find -size -10k
```



Important

The **-size** unit modifiers round everything up to single units. For example, **find -size 1M** will show files smaller than 1MB because it rounds all files up to 1MB.

The **-mmin** option, followed by the time in minutes, searches for all files that had their content changed at exactly the given time in the past.

To find all files that had their file content changed exactly 120 minutes ago on serverX, run:

Chapter14. Accessing Linux File Systems

```
[root@serverX ~]# find / -mmin 120
```

The **+** modifier in front of the amount of minutes looks for all files in the **/** that have been modified more than 200 minutes ago.

```
[root@serverX ~]# find / -mmin +200
```

The **-** modifier changes the search to look for all files in the **/** directory which have been changed less than 150 minutes ago.

```
[root@serverX ~]# find / -mmin -150
```

The **-type** option limits the search scope to a given file type, such as:

- **f**, for regular file
- **d**, for directory
- **l**, for soft link
- **b**, for block device

Search for all directories in the **/etc** folder on serverX.

```
[root@serverX ~]# find /etc -type d  
/etc  
/etc/tmpfiles.d  
/etc/systemd  
/etc/systemd/system  
/etc/systemd/system/getty.target.wants  
...
```

Search for all soft links on the serverX system.

```
[root@serverX ~]# find / -type l
```

Generate a list of all block devices in the **/dev** directory on serverX:

```
[root@serverX ~]# find /dev -type b  
/dev/vda1  
/dev/vda
```

The **-links** option followed by a number looks for all files that have a certain hard link count. The number can be preceded by a **+** modifier to look for files with a count higher than the given hard link count. If the number is preceded with a **-** modifier, the search is limited to all files with a hard link count that is less than the given number.

Search for all regular files with more than one hard link on the serverX machine:

```
[root@serverX ~]# find / -type f -links +1
```



References

locate(1), **updatedb(8)**, and **find(1)** man pages

Practice: Locating Files on the System

Guided exercise

In this lab, students will find files on the local file system.

Outcomes:

The user will search files with **locate** and **find**.

1. Use the **locate** command to find various different files on the serverX machine.

- 1.1. Even though the **locate** database is updated automatically every day, make sure the database is up-to-date by manually starting an update on serverX.

```
[root@serverX ~]# updatedb
```

- 1.2. Locate the configuration file **logrotate.conf** on serverX.

```
[root@serverX ~]# locate logrotate.conf
/etc/logrotate.conf
/usr/share/man/man5/logrotate.conf.5.gz
```

- 1.3. Locate the configuration file **networkmanager.conf**, ignoring case, on serverX.

```
[root@serverX ~]# locate -i networkmanager.conf
/etc/NetworkManager/NetworkManager.conf
/etc/dbus-1/system.d/org.freedesktop.NetworkManager.conf
/usr/share/man/man5/NetworkManager.conf.5.gz
```

2. Use the **find** command to perform real-time searches on the serverX machine according to the following requirements:

- 2.1. Find all files in the **/var/lib** directory owned by user **chrony** on serverX.

```
[root@serverX ~]# find /var/lib -user chrony
/var/lib/chrony
```

- 2.2. List all files in the **/var** directory owned by user **root** and group **mail**.

```
[root@serverX ~]# find /var -user root -group mail
/var/spool/mail
/var/spool/mail/root
```

- 2.3. List all files in the **/usr/bin** directory with a file size greater than 50 kilobytes.

```
[root@serverX ~]# find /usr/bin -size +50k
/usr/bin/pre-grohtml
/usr/bin/iconv
/usr/bin/localedef
/usr/bin/rpcgen
```

```
/usr/bin/less  
...
```

- 2.4. Find all files in the **/home/student** directory that have not been changed in the last 120 minutes on serverX.

```
[root@serverX ~]# find /home/student -mmin +120  
/home/student  
/home/student/.bash_logout  
/home/student/.bash_profile  
/home/student/.bashrc  
/home/student/.ssh  
...
```

- 2.5. Find all files in the **/tmp** directory that have been changed in the last 240 minutes on serverX.

```
[root@serverX ~]# find /tmp -mmin -240  
/tmp  
/tmp/.X11-unix  
/tmp/.X11-unix/X0  
/tmp/.ICE-unix  
...
```

Lab: Accessing Linux File Systems

Performance checklist

In this lab, students will mount a local file system, review it, and work with soft links.

Outcomes:

- Generate a disk usage report.
- Mount a file system.
- Create a soft link.
- Search files in the local file system.

Before you begin...

Reset your serverX system.

Run the **lab fs setup** to set up the server machine for your exercise.

1. Generate a disk usage report with the **du** command of the **/var/log** directory on serverX, and save the result in the **/tmp/results.txt** file.
2. Identify and mount a newly added file system by UUID on the **/mnt/myfreespace** directory on serverX.
3. Create the soft link **/root/myfreespace**, which points to the **/mnt/myfreespace** directory on serverX.
4. Find all soft links on serverX that have **freespace** as part of their names.

Solution

In this lab, students will mount a local file system, review it, and work with soft links.

Outcomes:

- Generate a disk usage report.
- Mount a file system.
- Create a soft link.
- Search files in the local file system.

Before you begin...

Reset your serverX system.

Run the **lab fs setup** to set up the server machine for your exercise.

1. Generate a disk usage report with the **du** command of the **/var/log** directory on serverX, and save the result in the **/tmp/results.txt** file.

```
[root@serverX ~]# du /var/log >/tmp/results.txt
```

2. Identify and mount a newly added file system by UUID on the **/mnt/myfreespace** directory on serverX.
 - 2.1. Identify the newly added file system with the **blkid** command on serverX.

```
[root@serverX ~]# blkid  
/dev/vda1: UUID="46f543fd-78c9-4526-a857-244811be2d88" TYPE="xfs"  
/dev/vdb1: UUID="a84f6842-ec1d-4f6d-b767-b9570f9fc0" TYPE="xfs"
```

- 2.2. Create the mount point **/mnt/myfreespace** on serverX.

```
[root@serverX ~]# mkdir /mnt/myfreespace
```

- 2.3. Mount the file system by UUID on the **/mnt/myfreespace** directory of the serverX machine.

```
[root@serverX ~]# mount UUID="a84f6842-ec1d-4f6d-b767-b9570f9fc0" /mnt/  
myfreespace
```

3. Create the soft link **/root/myfreespace**, which points to the **/mnt/myfreespace** directory on serverX.

```
[root@serverX ~]# ln -s /mnt/myfreespace /root/myfreespace
```

4. Find all soft links on serverX that have **freespace** as part of their names.

```
[root@serverX ~]# find / -type l -name '*freespace*'
```

```
/root/myfreespace
```

Summary

Identifying File Systems and Devices

Storage devices are represented by different device files.

Mounting and Unmounting File Systems

Accessing contents of file systems on internal and external storage devices is important.

Making Links Between Files

Handling links to existing files can save space on the file system.

Locating Files on the System

Searching for files is important for various administrative tasks.



CHAPTER 15

USING VIRTUALIZED SYSTEMS

Overview	
Goal	To create and use Red Hat Enterprise Linux virtual machines with Kernel-based Virtual Machine (KVM) and libvirt.
Objectives	<ul style="list-style-type: none">Install a Red Hat Enterprise Linux system as a host for running virtual machines.Perform an interactive install of Red Hat Enterprise Linux on a virtual machine.
Sections	<ul style="list-style-type: none">Managing a Local Virtualization Host (and Practice)Installing a New Virtual Machine (and Practice)
Chapter Test	<ul style="list-style-type: none">Using Virtualized Systems

Managing a Local Virtualization Host

Objectives

After completing this section, students should be able to:

- Describe and contrast Red Hat virtualization platforms.
- Install Red Hat Enterprise Linux as a virtualization host system.

System virtualization and Red Hat Enterprise Linux

KVM (Kernel-based Virtual Machine) is a full virtualization solution built into the standard Red Hat Enterprise Linux kernel. It can run multiple, unmodified Windows and Linux guest operating systems. The KVM hypervisor in Red Hat Enterprise Linux is managed with the *libvirt* API and utilities, such as **virt-manager** and **virsh**. Since Red Hat Enterprise Linux is the foundation of Red Hat Enterprise Virtualization and the Red Hat OpenStack platform, KVM is a consistent component across products of the Red Hat cloud infrastructure.

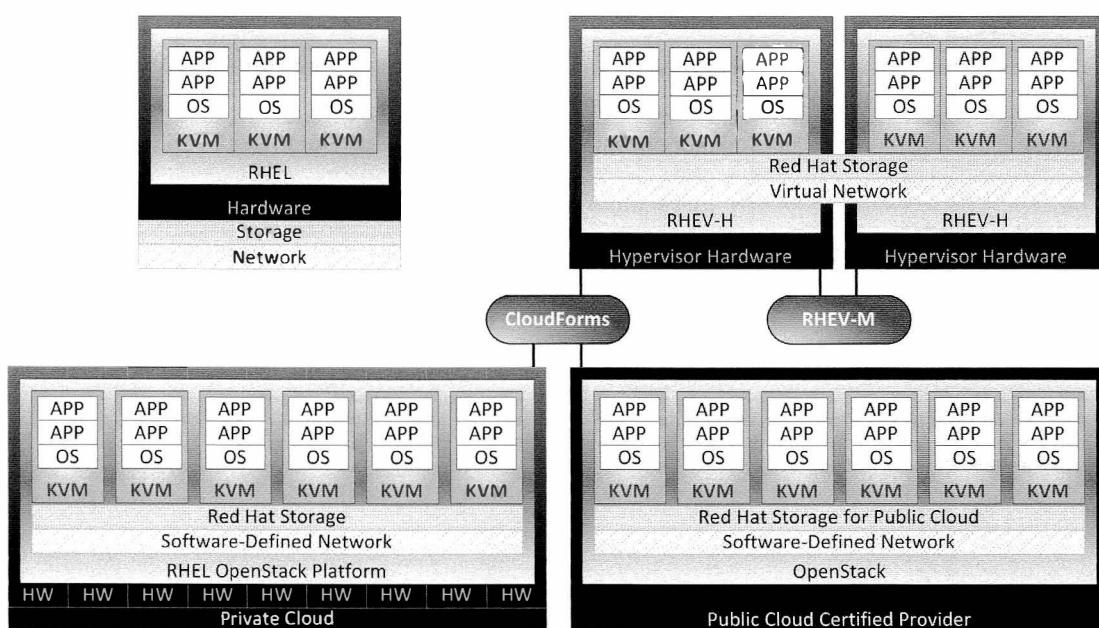


Figure 15.1: KVM throughout the Red Hat cloud infrastructure

KVM provides the virtual machine (VM) technology across all Red Hat products, from standalone physical instances of Red Hat Enterprise Linux up to the cloud OpenStack platform. Starting at the upper-left corner of the previous figure:

- **Physical (legacy) systems**—Red Hat Enterprise Linux installations on legacy hardware provide KVM virtualization, up to the physical limitations of single systems, and are managed by libvirt utilities such as **virt-manager**. Red Hat Enterprise Linux instances may also be directly hosted at Red Hat Certified Cloud Providers through Red Hat Cloud Access.

Red Hat Enterprise Linux is typically configured as a *thick host*, a system that supports VMs while also providing other local and network services, applications, and management functions.

- **Red Hat Enterprise Virtualization (RHEV)**—supports KVM instances across multiple Red Hat Enterprise Virtualization hypervisor (RHEV-H) systems, providing KVM migration, redundancy, and high availability managed by RHEV Manager (RHEV-M).

Red Hat Enterprise Virtualization Hypervisor is a *thin host*, an expertly minimized and tuned version of Red Hat Enterprise Linux dedicated to the singular purpose of provisioning and supporting guest VMs.

- **RHEL OpenStack platform**—Red Hat private cloud architecture using integrated and tuned OpenStack on a Red Hat Enterprise Linux foundation with KVM, managed by either the Red Hat OpenStack Dashboard (Horizon component) or by Red Hat CloudForms.
- **OpenStack in public cloud**—OpenStack public cloud architecture implemented at Red Hat Certified Cloud Providers, and managed by either the OpenStack Horizon component or by Red Hat CloudForms.
- **Hybrid Cloud**—Red Hat CloudForms cloud management utilities manage and migrate KVM instances across Red Hat RHEV and OpenStack architectures, and transition KVM instances with third-party OpenStack and VMware platforms.

KVM instance configurations are compatible across Red Hat products. Installation requirements, parameters, and procedures are equivalent on supported platforms.

Configure a Red Hat Enterprise Linux Physical System as a Virtualization Host

Red Hat Enterprise Linux can be configured as a virtualization host, appropriate for development, testing, training, or when needing to work in multiple operating systems at the same time. Red Hat Enterprise Linux hosts provide the ability to install additional software on the host platform as needed, such as monitoring utilities and agents, network services, specialized storage, and/or other development tools that may not be appropriate to install on dedicated Red Hat Enterprise Virtualization hypervisors.

Red Hat Enterprise Linux installations also provide easier access to tuning and resource management tools (e.g., **tuned** and **cgroups**). By comparison, RHEV-H hypervisors are highly secured and self-tuned, restricting system administrator-initiated customization by design. When greater administrative control is required and the performance compromise is acceptable, Red Hat Enterprise Linux is a flexible standalone KVM platform. KVM instances built on RHEL can be migrated or transitioned to more appropriate KVM platforms as enterprise needs increase.

Preparing a Red Hat Enterprise Linux system to become a virtualization host requires checking for minimal system requirements and installing a choice of virtualization host packages.

Recommended system requirements:

- One processor core or hyperthread for the maximum number of virtualized CPUs in a guest virtual machine and one for the host.
- 2GB of RAM, plus additional RAM for virtual machines.
- 6GB disk space for the host, plus the required disk space for each virtual machine. Most guest operating systems require at least 6GB of disk space, but actual storage space requirements depend on each guest's image format.

The KVM hypervisor requires either an Intel processor with the Intel VT-x and Intel 64 extensions for x86-based systems, or an AMD processor with the AMD-V and the AMD64 extensions. To verify that the host system hardware supports the correct extensions, view **/proc/cpuinfo**.

```
[root@serverX ~]# grep --color -E "vmx|svm" /proc/cpuinfo
flags          : fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov pat
pse36 clflush dts acpi mmx fxsr sse sse2 ss ht tm pbe syscall nx lm constant_tsc
arch_perfmon pebs bts rep_good aperfmpf perf_pni dtes64 monitor ds_cpl vmx smx est
tm2 ssse3 cx16 xtpr pdcm sse4_1 xsave lahf_lm dts tpr_shadow vnmi flexpriority
```

The No eXecute (NX) feature, called eXecute Disable (XD) by Intel and Enhanced Virus Protection by AMD, is not necessary for building a host on Red Hat Enterprise Linux, but is required for a Red Hat Enterprise Virtualization hypervisor (RHEV-H).

```
[root@serverX ~]# grep --color -E "nx" /proc/cpuinfo
flags          : fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov pat
pse36 clflush dts acpi mmx fxsr sse sse2 ss ht tm pbe syscall nx lm constant_tsc
arch_perfmon pebs bts rep_good aperfmpf perf_pni dtes64 monitor ds_cpl vmx smx est
tm2 ssse3 cx16 xtpr pdcm sse4_1 xsave lahf_lm dts tpr_shadow vnmi flexpriority
```

Building a RHEL virtualization host requires the **qemu-kvm** and **qemu-img** packages at minimum, to provide the user-level KVM emulator and disk image manager.

```
[root@serverX ~]# yum install qemu-kvm qemu-img
```

Additional virtualization management packages are also recommended:

- **python-virtinst**—Provides the `virt-install` command for creating virtual machines.
- **libvirt**—Provides the host and server libraries for interacting with hypervisors and host systems.
- **libvirt-python**—Contains a module to permit Python applications to use the libvirt API.
- **virt-manager**—Provides the Virtual Machine Manager graphical tool for administering VMs, using the libvirt-client library as the management API.
- **libvirt-client**—Provides the client APIs and libraries for accessing libvirt servers, including the `virsh` command-line tool to manage and control VMs.

```
[root@serverX ~]# yum install virt-manager libvirt libvirt-python python-virtinst
libvirt-client
```

The updated **anaconda** graphical installation program for Red Hat Enterprise Linux 7 provides better support for installing RHEL to meet specific purposes. An **anaconda** install no longer provides the ability to select individual RPM packages—only base environments and add-ons appropriate for the selected base—thus eliminating guesswork and resulting in leaner configurations. System administrators may still install any other desired RPM packages after an installation is complete by using standard RPM installation tools (e.g., **yum** or GNOME PackageKit).

To build a virtualization host during a graphical install of Red Hat Enterprise Linux, choose the base environment **Virtualization Host** in the left pane of the **anaconda Software Selection** screen. Select the **Virtualization Platform Add-On** checkbox in the right pane to include the management tools and utilities, as shown in the following figure.

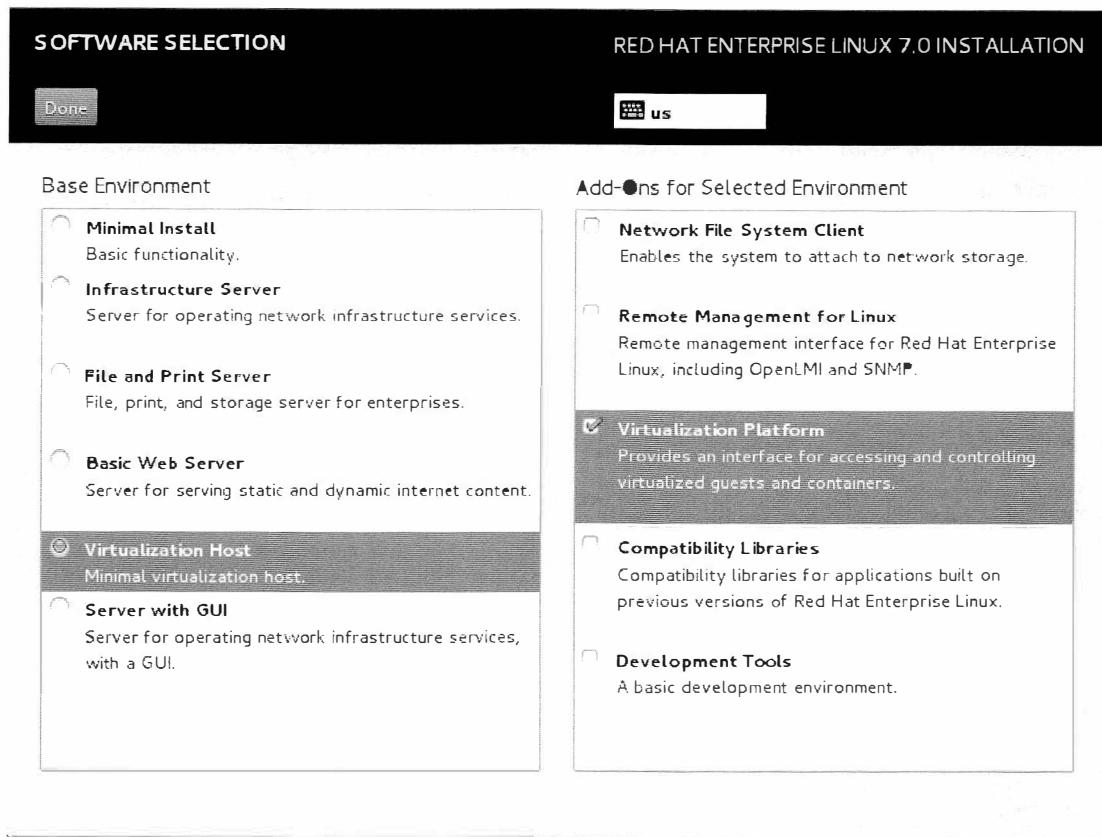


Figure 15.2: Creating a virtualization host during a graphical installation

Managing virtual machines

The libvirt package is a hypervisor-independent virtualization API to securely manage virtual machines by providing the ability to provision, create, modify, monitor, control, migrate, and stop virtual machines on a single host. The libvirt package provides APIs to enumerate, monitor, and use the resources available on the managed host, including CPUs, memory, storage, and networking. Management tools utilizing libvirt can access host systems remotely using secure protocols.

Red Hat Enterprise Linux uses libvirt-based tools as the default for virtualization management. Support is included for the RHEL 5 Xen hypervisor, and for KVM on RHEL 5, 6, and 7. These management tools utilize libvirt:

- **virsh**—The virsh command-line tool is an alternative to the graphical virt-manager application. Unprivileged users can use virsh in read-only mode, or with root access for full administrative functionality. The virsh command is ideal for scripting virtualization administration.
- **virt-manager**—virt-manager is a graphical desktop tool allowing access to guest consoles, and is used to perform virtual machine creation, migration, configuration, and administrative tasks. Both local and remote hypervisors can be managed through a single interface.
- **RHEV-M**—Red Hat Enterprise Virtualization Manager provides a central management platform for physical and virtual resources, allowing virtual machines to be started, stopped, built, and migrated between hosts. RHEV-M also manages the storage and network components of a data center and provides secure, remote graphical guest console access.

Start the Virtual Machine Manager from the menu at **Applications > System Tools > Virtual Machine Manager**, or by running the **virt-manager** command from the shell. Use this interface to power on or power off virtual machines, assign memory and CPU resources, monitor performance, and connect to the console of the virtual machines.

The **virsh** command-line tool provides the same functionality as **virt-manager**. Use **virsh** as an interactive shell to perform subcommands such as edit, list, start, stop, and destroy. The following examples show the **virsh** commands run as standalone commands from the shell:

```
[root@foundationX ~]# virsh list
  Id  Name      State
  -- -
  1  desktop    running
  2  server    running

[root@foundationX ~]# virsh destroy server
[root@foundationX ~]# virsh list --all
  Id  Name      State
  -- -
  1  desktop    running
 - server    shut off

[root@foundationX ~]# virsh start server
[root@foundationX ~]# virsh list
  Id  Name      State
  -- -
  1  desktop    running
  2  server    running
```

virsh has subcommands for additional management tasks:

- connect—Connect to a local or remote KVM host using **qemu:///host** syntax.
- nodeinfo—Returns basic information about the host, including CPUs and memory.
- autostart—Configures a KVM domain to start when the host boots.
- console—Connect to the virtual *serial* console of a guest.
- create—Create a domain from an XML configuration file and start it.
- define—Create a domain from an XML configuration file, but do not start it.
- undefine—Undefine a domain. If the domain is inactive, the domain configuration is removed.
- edit—Edit the XML configuration file for a domain, which will affect the next boot of the guest.
- reboot—Reboot the domain, as if the **reboot** command had been run from inside the guest.
- shutdown—Gracefully shuts down the domain, as if the **shutdown** command had been run from inside the guest.
- screenshot—Takes a screenshot of a current domain console and stores it in a file.



References

Additional information may be available in the introduction and chapter on system requirements in the *Red Hat Enterprise Linux Virtualization Deployment and Administration Guide* for Red Hat Enterprise Linux 7, which can be found at
<http://docs.redhat.com/>

Red Hat Enterprise Virtualization Administration Guide

- Section 1. Basics

Red Hat Enterprise Linux OpenStack Platform 4 Getting Started Guide

- Section 1. Introduction

virsh(1), virt-manager(1) man pages

Practice: Managing a Local Virtualization Host

Quiz

Match the following items to their counterparts in the table.

create	define	destroy	reboot	shutdown	start

Purpose	virsh subcommand
Boot an existing configured virtual machine	
Immediately stop a virtual machine, similar to unplugging it	
Delete the configuration for a virtual machine permanently	
Use an XML configuration to create and boot a virtual machine	
Use an XML configuration to create a virtual machine	
Gracefully stop and restart a virtual machine	
Gracefully stop a virtual machine	

Solution

Match the following items to their counterparts in the table.

Purpose	virsh subcommand
Boot an existing configured virtual machine	start
Immediately stop a virtual machine, similar to unplugging it	destroy
Delete the configuration for a virtual machine permanently	undefine
Use an XML configuration to create and boot a virtual machine	create
Use an XML configuration to create a virtual machine	define
Gracefully stop and restart a virtual machine	reboot
Gracefully stop a virtual machine	shutdown

Installing a New Virtual Machine

Objectives

After completing this section, students should be able to:

- Build a virtual machine configuration.
- Install Red Hat Enterprise Linux into a new virtual machine instance.

Creating a virtual machine

Virtual machines can perform the same tasks as physical systems. System administrators make system sizing and configuration decisions using fundamentally the same criteria as for physical systems, including the guest machine's role and projected system load. Preparatory considerations must include CPU and memory requirements, type of I/O and the expected number of clients, access to public or exclusive storage, current and future sizing expectations, and bandwidth and latency requirements. Necessary disk and network components must be configured on the virtual host before creating the guest.

Launch the Virtual Machine Manager from the menu **Applications > System Tools > Virtual Machine Manager**, or run the **virt-manager** command as root. Click the **Create a new virtual machine** button to open the **New VM** wizard. Ensure that virt-manager can access the installation media (whether locally or over the network) before continuing.

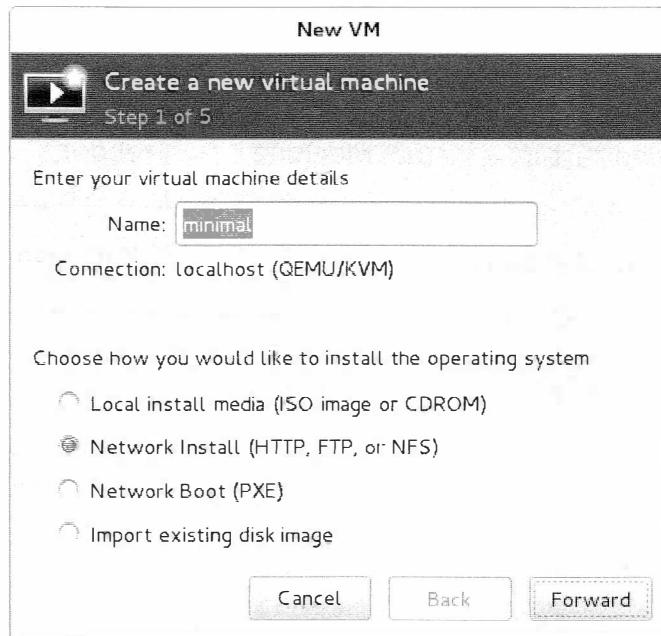


Figure 15.3: Create a virtual machine (step 1 of 5)

Select a name for the virtual machine, which is used as the configuration domain name. The system host name is configured later during the operating system installation. Choices for installation type depend on what resources are prepared. An operating system DVD must be available on the virtual host or on physical media before selecting **Local install media**, for

example, or a network installation server must be available when selecting **Network Boot (PXE)**. Sharing and accessing the installation media using any file sharing protocol is chosen here.

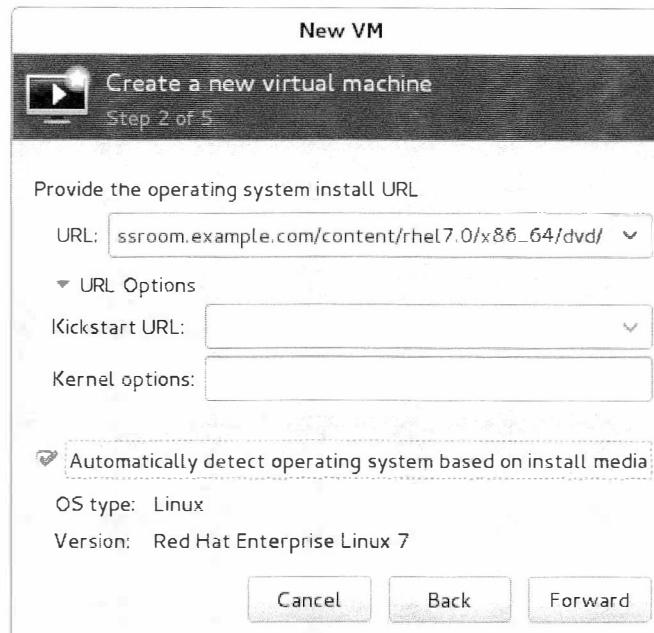


Figure 15.4: Create a virtual machine (step 2 of 5)

Enter the URL for the installation media network resource. Configure an appropriate **OS type** and **Version** for this guest. When the installation media is currently accessible, selecting **Automatically detect operating system** fills these fields. If the fields remain blank or are incorrect, fix the installation media before continuing. The **Kickstart** options allow unattended installations; a Kickstart server and preconfigured client configuration file are required.

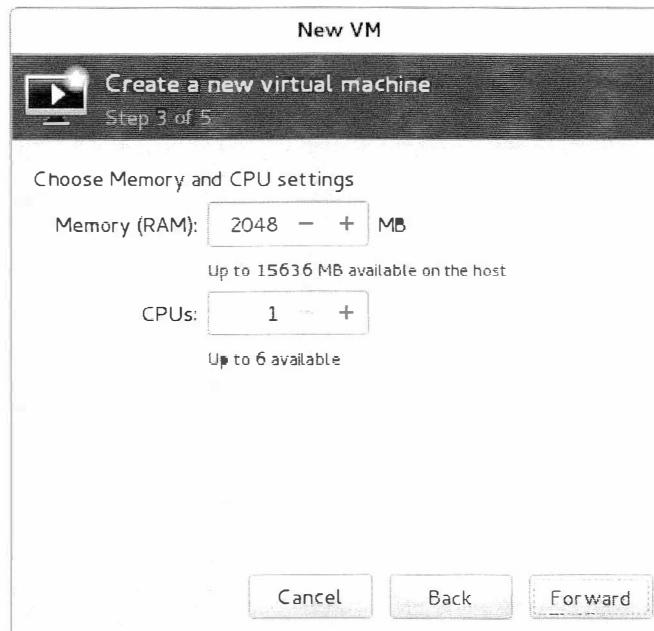


Figure 15.5: Create a virtual machine (step 3 of 5)

Configure an appropriate amount of **Memory** and **CPUs**. Red Hat Enterprise Linux 7 requires a minimum of one CPU and 1024MB of memory.

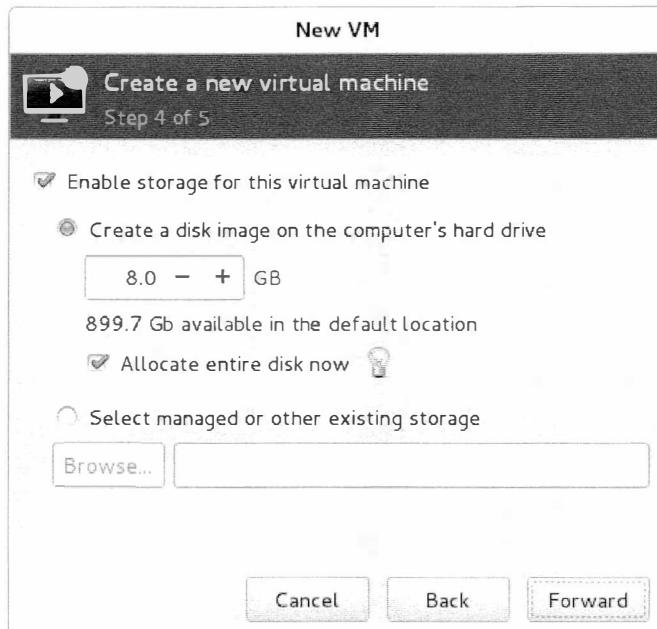


Figure 15.6: Create a virtual machine (step 4 of 5)

Assign storage to the guest virtual machine. For a created disk image, allocating now will provide a small performance benefit, while allocating later saves disk space not yet needed. To **Select managed or other existing storage** requires having previously prepared a disk image, physical device, or logical volume prior to beginning this install.

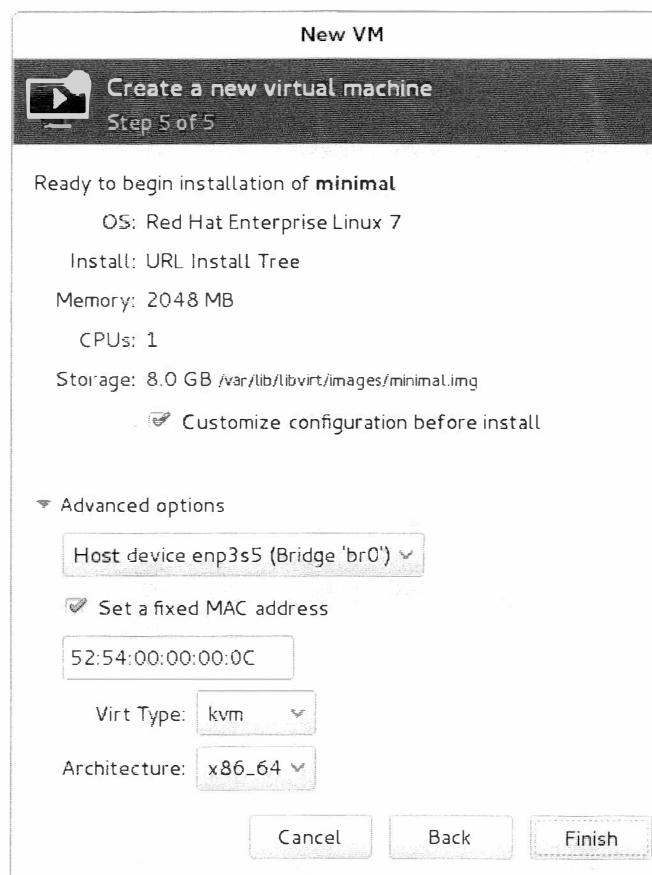


Figure 15.7: Create a virtual machine (step 5 of 5)

Verify the settings of the virtual machine in the top half of the dialog window. Open the **Advanced options** to configure networking. Virtual machines can be configured as private guests using *Network Address Translation* (NAT), or given the appearance of direct subnet access by using the virtual host's preconfigured bridge.

The **New VM** wizard automatically generates a MAC address in the 52:54:00 range for virtual guests. Use the random MAC provided or replace the address with one that fits the environment requirements. Unless the **Customize configuration before install** checkbox is selected, the virtual machine creation will begin when the **Finish** button is pressed.



Note

In production environments, DNS and DHCP databases are preconfigured with host names and reserved IP addresses assigned to registered MAC addresses so as to centralize and coordinate enterprise-wide virtual system management. In a Red Hat Training environment, the classroom DNS server has been populated with MAC addresses and unique names for anticipated student guest systems. Use an appropriate MAC address as directed by the instructor or exercise instructions.

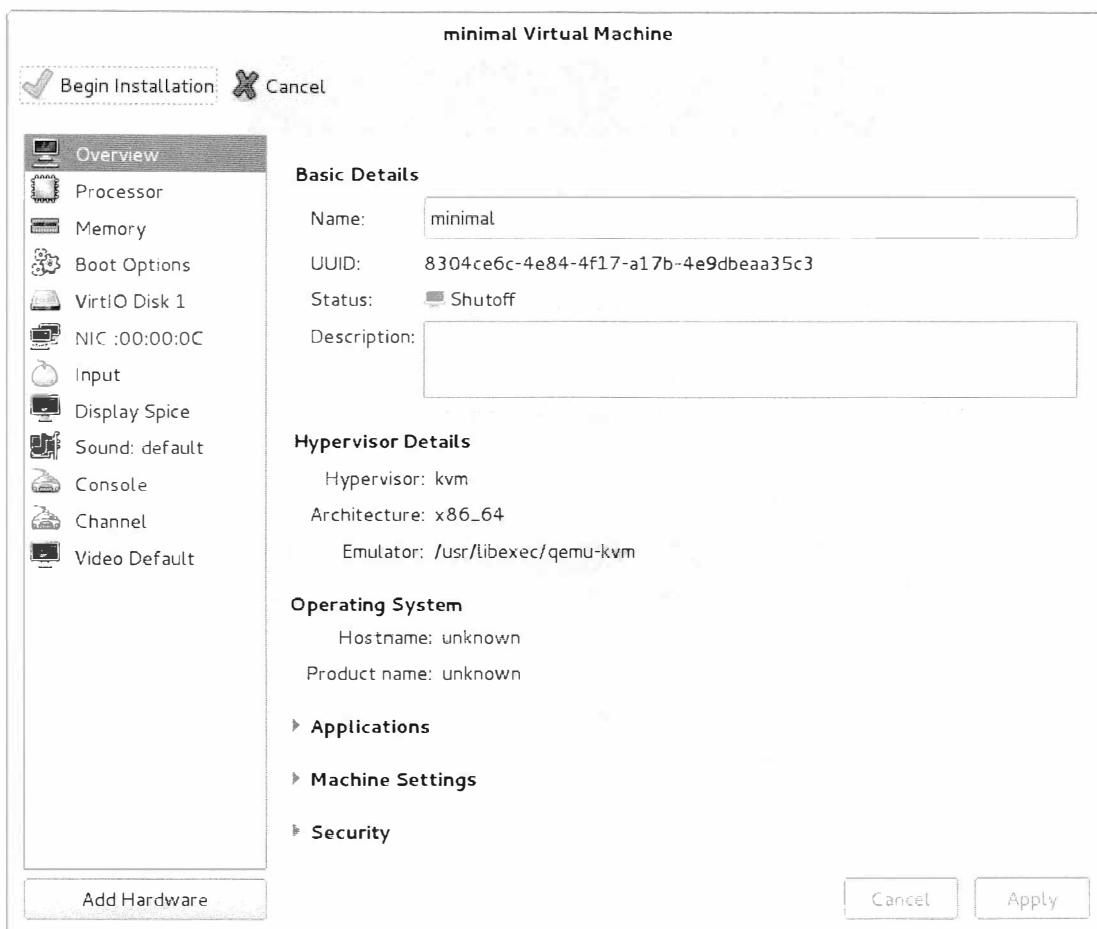


Figure 15.8: Create a virtual machine, customize configuration

If the **Customize configuration before install** checkbox was selected on the last step of the **New VM** wizard, the **domain_name Virtual Machine** detail screen displays. Administrators can further modify or correct the configuration before the installation begins. Additional hardware—such as graphics and other add-in cards, disks, and network interfaces—may be configured here.

When customization is complete, press **Begin Installation** in the upper-left corner.

When the installation is complete, administrators can return to this same detail screen to modify or correct the virtual machine configuration. The option for automatically starting this virtual machine when the physical host boots is also found here.

Installing Red Hat Enterprise Linux

It is recommended that Red Hat Enterprise Linux be installed using the graphical interface, known as **anaconda**. If anaconda detects that an installation started in text mode where a VNC connection might be possible, anaconda asks to verify. The text mode installation is simpler, but certain options available in graphical mode are not available in text mode.

During the installation, virtual consoles accessed from the physical machine provide information such as diagnostic messages and the ability to enter commands from a shell prompt. This capability is not available during a remote installation. The following table lists the virtual consoles available during installation, and the keystrokes used to switch between them.

Description of virtual consoles

Console	Keyboard shortcut	Contents
1	Ctrl+Alt+F1	Main installer console, contains debugging information from anaconda
2	Ctrl+Alt+F2	Shell prompt with root access, contains a stored history of useful commands
3	Ctrl+Alt+F3	Installation log, displays messages stored in /tmp/anaconda.log
4	Ctrl+Alt+F4	Storage log, displays messages on related storage devices from kernel and system services, stored in /tmp/storage.log
5	Ctrl+Alt+F5	Program log, displays messages from other system utilities, stored in /tmp/program.log
6	Ctrl+Alt+F6	Spare shell prompt
7	Ctrl+Alt+F7	The default console with GUI

At the **Welcome to Red Hat Enterprise Linux 7.0** screen, select the language to be used during installation. After install, users select their own preferred language upon new account login.

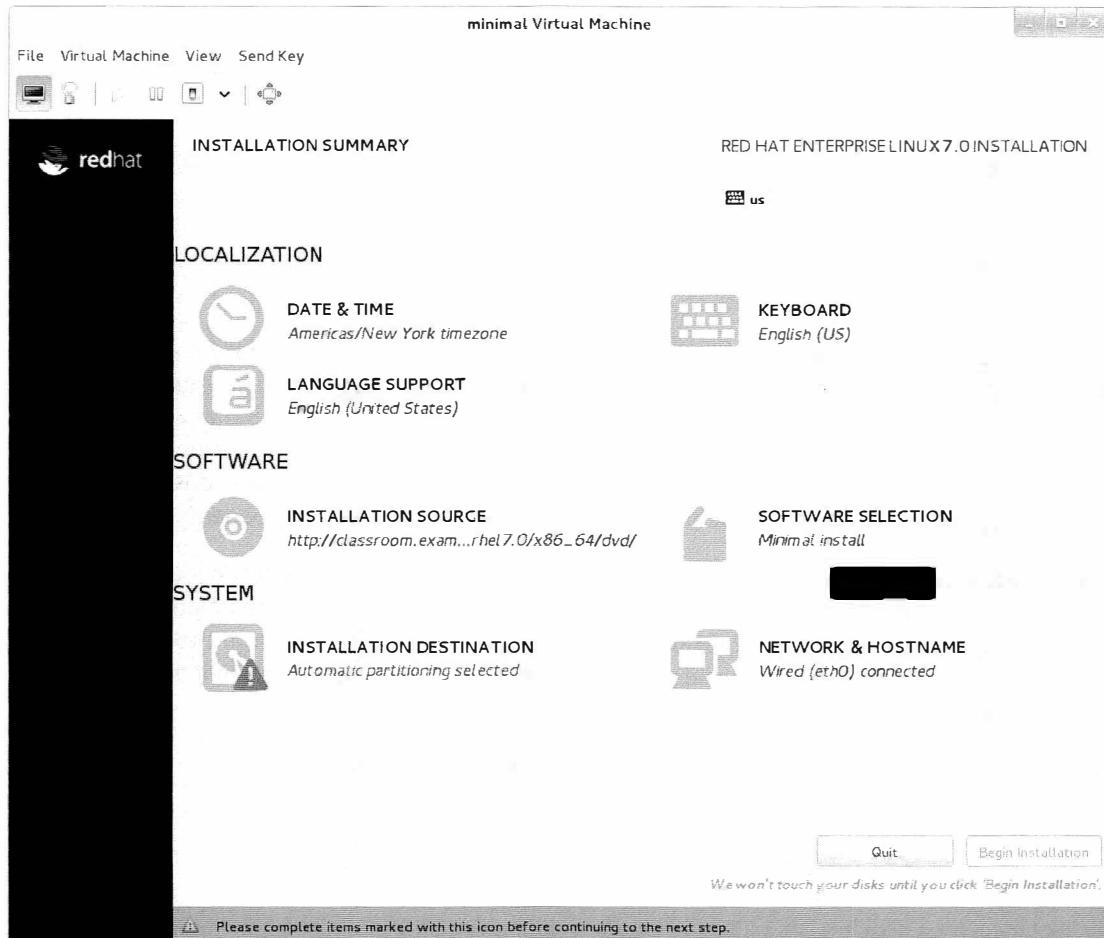


Figure 15.9: Anaconda installer, installation summary

The **anaconda Installation Summary** screen displays. From this central screen, installation customizations may be provided before beginning the installation. The installer allows configuring the installation items in any order. Select an item to view or edit. When an item is completed, or it is intended to be completed later, press **Done** to return to this central screen.

Only items marked with a warning symbol are mandatory. The orange status bar at the bottom of the screen warns that these items must be completed before the installation can begin. Once required items are complete, press the **Begin Installation** button. Pressing the **Quit** button aborts the installation. The virtual machine configuration will remain, but be unbootable until an operating system installation is restarted and completed.

As required, complete the following items:

- **Date and time** - Select your city by clicking in the interactive map or select from the dropdown list. Specify a time zone even when using Network Time Protocol (NTP). Messages complaining that NTP is not configured may be ignored if no NTP servers are available.
- **Language support** - Select languages to install in addition to the default language already specified. Multiple languages and locales may be selected.
- **Keyboard** - This item allows adding additional keyboard layouts beyond that included with your default language.
- **Installation source** - The installation source was selected during the virtual machine creation steps and should not need to be modified.
- **Software selection** - By default, the graphical installer selects the Minimal install environment, providing only packages essential to run Red Hat Enterprise Linux. In Software Selection, choose from a list of other base environments by clicking the radio button for an available environment listed in the left pane. Next, select Add-Ons for the selected environment from the checkboxes in the right pane.
- **Installation destination** - Select and partition the disks onto which Red Hat Enterprise Linux will install. This item expects an administrator to comprehend partitioning schemes and file system selection criteria. The default radio button for automatic partitioning will allocate the selected storage devices using all available space. If there are no other operating systems already installed on this computer or you have chosen not to preserve a previously installed operating systems, **anaconda** automatically installs GRUB2 as the boot loader.
- **Network & host name** - Detected network connections are listed in the left pane. Click a listed connection to display more details. To configure a network connection manually, click the **Configure** button in the lower-right corner. A dialog appears to configure the selected connection. Configuration options depend on available network hardware. Only network connections required for installation must be configured here.

When installation customization is complete, press **Begin Installation**.

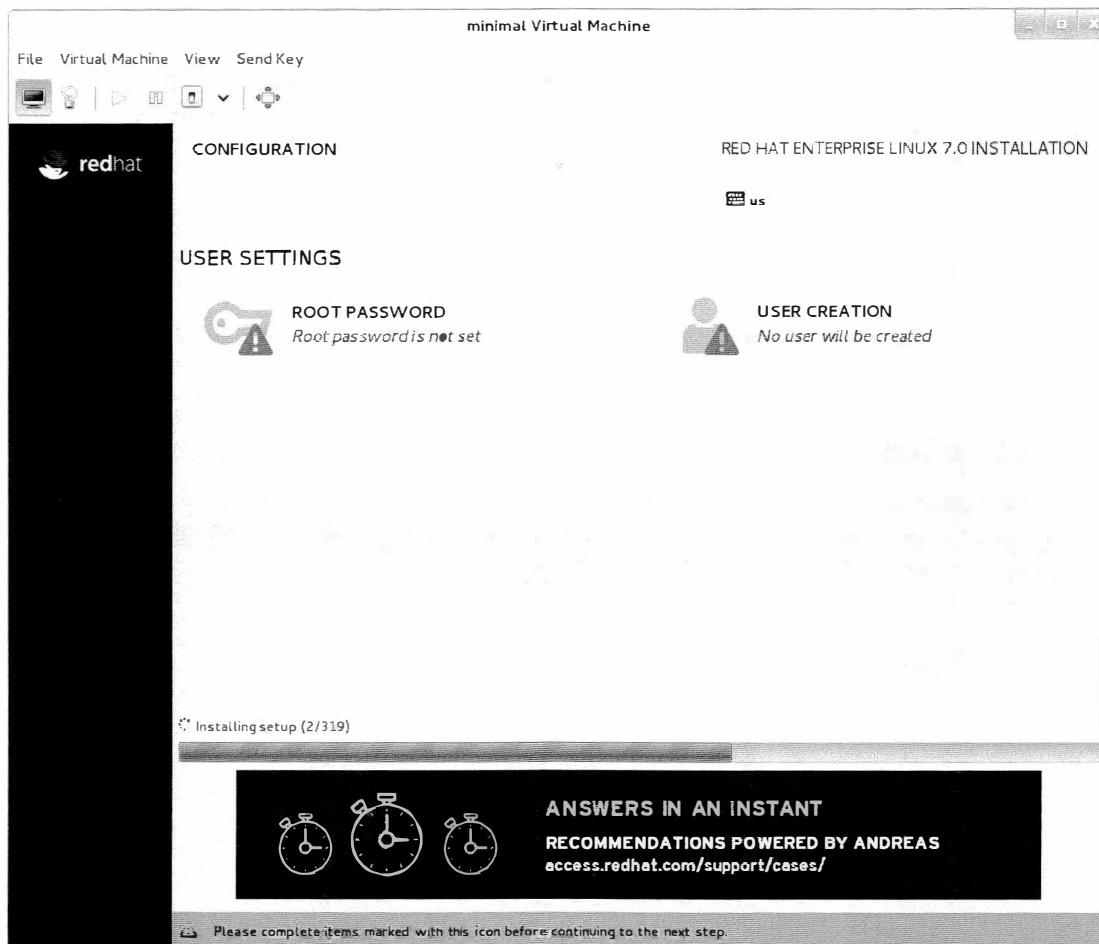


Figure 15.10: Anaconda installer, configuration

The **anaconda Configuration** screen displays. The installation has already started; Red Hat Enterprise Linux reports progress near the bottom of the screen as it installs selected packages to the system.

As required, complete the following items:

- **Root password** - The installation program prompts to set a root password for the system. The final stage of the installation process will not continue until a root password is entered.
- **User creation** - Use the root account only for system administration. Create a non-root account for general use. Though recommended to do during installation, this step is optional and can be performed after the installation is complete.

When the installation states that it is complete, press **Reboot**. After the virtual machine's normal power-up sequence has completed, Red Hat Enterprise Linux loads and starts, hidden behind a graphical screen that displays a progress bar. If a graphical desktop was installed, the GUI login displays. Log in as the user created during installation, or as root. The **anaconda Initial Setup** screen displays.

As required, complete the following items:

- **License information** - The installation program prompts to accept the licensing terms.

- **User creation** - Displays only if a non-root account has not yet been created.

When initial setup tasks are complete, press **Finish Installation**. The **FirstBoot** utility displays, prompting entry of the final configuration information for this system. Use this utility to:

- Configure the Kdump crash dump mechanism.
- Set the system time and date.
- Register the machine with Red Hat Network through Subscription Management.

The Red Hat Enterprise Linux installation is now complete. The system is ready for normal login and use.



References

Additional information may be available in the section on creating guests with virt-manager in the *Red Hat Enterprise Linux Virtualization Deployment and Administration Guide* for Red Hat Enterprise Linux 7, which can be found at

<http://docs.redhat.com/>

virt-manager(1) man page

Practice: Installing a New Virtual Machine

Quiz

The steps to install Red Hat Enterprise Linux using the **anaconda** graphical interface are shown below. Indicate the order the steps should be taken.

- a. Provide the source location and operating system type
- b. Modify Localization parameters for Date and Time, Language and Keyboard
- c. Provide System parameters for disk partitioning, networking and hostname
- d. Configure storage for this virtual machine
- e. Enter the Memory and CPU settings
- f. Provide User Settings for a root password and a non-root account
- g. Name the virtual machine and select an installation source.
- h. Select the Software to be installed
- i. Configure networking options for the installation

Solution

The steps to install Red Hat Enterprise Linux using the **anaconda** graphical interface are shown below. Indicate the order the steps should be taken.

- 2 a. Provide the source location and operating system type
- 6 b. Modify Localization parameters for Date and Time, Language and Keyboard
- 8 c. Provide System parameters for disk partitioning, networking and hostname
- 4 d. Configure storage for this virtual machine
- 3 e. Enter the Memory and CPU settings
- 9 f. Provide User Settings for a root password and a non-root account
- 1 g. Name the virtual machine and select an installation source.
- 7 h. Select the Software to be installed
- 5 i. Configure networking options for the installation

Chapter Test: Using Virtualized Systems

Quiz

Match the following items to their counterparts in the table.

CloudForms	OpenStack in public cloud	
RHEL OpenStack platform	RHEV-M	Red Hat Enterprise Linux
Red Hat Enterprise Virtualization	virt-manager	

Configuration description	Red Hat product
Single system hardware providing KVM support	
Multiple system hardware providing virtualized redundancy	
Multiple system hardware providing private cloud	
Cloud provider providing public cloud	
Management utility for standalone KVM hosts	
Management utility for multiple host virtualization platform	
Management utility for all virtualization and cloud platforms combined	

Solution

Match the following items to their counterparts in the table.

Configuration description	Red Hat product
Single system hardware providing KVM support	Red Hat Enterprise Linux
Multiple system hardware providing virtualized redundancy	Red Hat Enterprise Virtualization
Multiple system hardware providing private cloud	RHEL OpenStack platform
Cloud provider providing public cloud	OpenStack in public cloud
Management utility for standalone KVM hosts	virt-manager
Management utility for multiple host virtualization platform	RHEV-M
Management utility for all virtualization and cloud platforms combined	CloudForms

Summary

Managing a Local Virtualization Host

Prepare and build virtualization infrastructure using Red Hat Enterprise Linux.

Installing a New Virtual Machine

Build virtual system components and install Red Hat Enterprise Linux into KVM virtual machines.



CHAPTER 16

COMPREHENSIVE REVIEW

Overview	
Goal	To practice and demonstrate knowledge and skills learned in Red Hat System Administration I.
Objectives	<ul style="list-style-type: none">• Practice skills learned in Red Hat System Administration I.
Lab	<ul style="list-style-type: none">• Comprehensive Review

Red Hat System Administration I

Comprehensive Review

Objectives

After completing this section, students should be able to demonstrate knowledge and skill of the topics covered in each chapter.

Reviewing Red Hat System Administration I

Before beginning the comprehensive review for this course, students should be comfortable with the topics covered in each chapter.

Do not hesitate to ask the instructor for extra guidance or clarification on these topics.

Chapter 1, Accessing the Command Line

To log into a Linux system and run simple commands using the shell.

- Use Bash shell syntax to enter commands at a Linux console.
- Launch applications in a GNOME desktop environment.
- Use Bash features to run commands from a shell prompt using fewer keystrokes.

Chapter 2, Managing Files From the Command Line

To copy, move, create, delete, and organize files while working from the Bash shell prompt.

- Identify the purpose for important directories on a Linux system.
- Specify files using absolute and relative path names.
- Create, copy, move, and remove files and directories using command-line utilities.
- Match one or more file names using shell expansion as arguments to shell commands.

Chapter 3, Getting Help in Red Hat Enterprise Linux

To resolve problems by using on-line help systems and Red Hat support utilities.

- Use the man Linux manual reader.
- Use the pinfo GNU Info reader.
- Use the Red Hat Package Manager (RPM) package documentation.
- Use the redhat-support-tool command.

Chapter 4, Creating, Viewing, and Editing Text Files

To create, view, and edit text files from command output or in an editor.

- Redirect the text output of a program to a file or to another program.
- Edit existing text files and create new files from the shell prompt with a text editor.
- Copy text from a graphical window to a text file using a text editor running in the graphical environment.

Chapter 5, Managing Local Linux Users and Groups

To manage local Linux users and groups and administer local password policies.

- Explain the role of users and groups on a Linux system and how they are understood by the computer.
- Run commands as the superuser to administer a Linux system.
- Create, modify, lock, and delete locally defined user accounts.
- Create, modify, and delete locally defined group accounts.
- Lock accounts manually or by setting a password-aging policy in the shadow password file.

Chapter 6, Controlling Access to Files with Linux File System Permissions

To set Linux file system permissions on files and interpret the security effects of different permission settings.

- Explain how the Linux file permissions model works.
- Change the permissions and ownership of files using command-line tools.
- Configure a directory in which newly created files are automatically writable by members of the group which owns the directory, using special permissions and default umask settings.

Chapter 7, Monitoring and Managing Linux Processes

To evaluate and control processes running on a Red Hat Enterprise Linux system.

- List and interpret basic information about processes running on the system.
- Control processes in the shell's session using bash job control.
- Terminate and control processes using signals.
- Monitor resource usage and system load due to process activity.

Chapter 8, Controlling Services and Daemons

To control and monitor network services and system daemons using systemd.

- List system daemons and network services started by the systemd service and socket units.
- Control system daemons and network services using `systemctl`.

Chapter 9, Configuring and Securing OpenSSH Service

To configure secure command-line access on remote systems using OpenSSH.

- Log into a remote system using ssh to run commands from a shell prompt.
- Set up ssh to allow secure password-free logins by using a private authentication key file.
- Customize sshd configuration to restrict direct logins as root or to disable password-based authentication.

Chapter 10, Analyzing and Storing Logs

To locate and accurately interpret relevant system log files for troubleshooting purposes.

- Describe the basic syslog architecture in Red Hat Enterprise Linux 7.
- Interpret entries in relevant syslog files to troubleshoot problems or review system status.

- Find and interpret log entries in the systemd journal to troubleshoot problems or review system status.
- Configure systemd-journald to store its journal on disk rather than in memory.
- Maintain accurate time synchronization and time zone configuration to ensure correct timestamps in system logs.

Chapter 11, Managing Red Hat Enterprise Linux Networking

To configure basic IPv4 networking on Red Hat Enterprise Linux systems.

- Explain fundamental concepts of computer networking.
- Test and review current network configuration with basic utilities.
- Manage network settings and devices with **nmcli** and NetworkManager.
- Modify network settings by editing the configuration files.
- Configure and test system host name and name resolution.

Chapter 12, Archiving and Copying Files Between Systems

To archive and copy files from one system to another.

- Use tar to create new compressed archive files and extract files from existing archive files.
- Copy files securely to or from a remote system running sshd.
- Securely synchronize the contents of a local file or directory with a remote copy.

Chapter 13, Installing and Updating Software Packages

To download, install, update, and manage software packages from Red Hat and YUM package repositories.

- Register systems with your Red Hat account and entitle them to software updates for installed products.
- Explain what an RPM package is and how RPM packages are used to manage software on a Red Hat Enterprise Linux system.
- Find, install, and update software packages using the **yum** command.
- Enable and disable use of Red Hat or third-party YUM repositories.
- Examine and install downloaded software package files.

Chapter 14, Accessing Linux File Systems

To access and inspect existing file systems on a Red Hat Enterprise Linux system.

- Identify the file system hierarchy.
- Access the contents of file systems.
- Use hard links and symlinks to make multiple names.
- Search for files on mounted file systems.

Chapter 15, Using Virtualized Systems

To create and use Red Hat Enterprise Linux virtual machines with Kernel-based Virtual Machine (KVM) and libvirt.

- Install a Red Hat Enterprise Linux system as a host for running virtual machines.
- Perform an interactive install of Red Hat Enterprise Linux on a virtual machine.

R References

Get information on more classes available from Red Hat at
www.redhat.com/training/

Lab: Comprehensive Review

Performance checklist

In this lab, you will practice and demonstrate your knowledge and skills.

Outcomes:

Complete the following tasks and successfully grade the serverX system with **lab sa1-review grade** as user root on serverX.

Before you begin...

Reset the serverX machine.

Run the **lab sa1-review setup** as user root on serverX.

1. Use Bash commands to complete the following tasks on the serverX machine:
 - Display the first 12 lines of the **/usr/bin/clean-binary-files** file and send the output to the **/home/student/headtail.txt** file.
 - Display the last nine lines of the **/usr/bin/clean-binary-files** file and add the output to the **/home/student/headtail.txt** file.
2. Ten new Linux systems require change documentation files. Carry out the following tasks on serverX to create them:
 - Create the empty files with the file name **system_changes-machineY-month_Z.txt** in the **/home/student** directory on the serverX machine as user student. Replace Y with the machine number and replace Z with the months *jan*, *feb*, and *mar*.
 - Create the **/home/student/syschanges** directory with the subdirectories **jan**, **feb**, and **mar**.
 - Sort all newly created files by month into the corresponding subdirectory.
 - Remove all newly created files related to machine 9 and 10, because the hardware has been replaced permanently.
3. Use the man pages to research how to turn off the use of colors in the output. Put the relevant option of the **ls** command into the text file **/home/student/lscolor.txt** on serverX.
4. Copy the file **/home/student/vimfile.txt** to **/home/student/longlisting.txt** on serverX. Use the **vim** editor to change the **/home/student/longlisting.txt** file according to the following requirements:
 - Remove the file owner column. Do not remove any spaces.
 - Remove the **Documents** and **Pictures** rows.
 - Save the file when done with editing.
5. Change configuration and add new users and a new group according to the following requirements:

- Change the default system settings for newly created users to ensure their passwords are changed at least every 60 days.
 - Create a new group named **instructors** with a GID of 30000.
 - Create three new users: **gorwell**, **rbradbury**, and **dadams**, with a password of **firstpw**.
 - Add the new users to the supplementary group **instructors**. The primary group should remain as the user private group.
 - Set the three newly created accounts to expire 60 days from today.
 - Change the password policy for the **gorwell** account to require a new password every 10 days.
 - Force all three newly created users to change their password on first login.
6. Create the shared directory **/home/instructors** on serverX according to the following requirements:
- The directory is owned by user root and group instructors.
 - Set permissions on the **/home/instructors** directory so it has the GID bit set on the directory, the owner and group have full read/write/execute permissions, and other users have read permission to the directory.
7. Determine the process using the most CPU resources on serverX and terminate it.
8. Stop the currently running cups printing service on serverX. The service should not get automatically started on system boot.
9. Configure the ssh service on serverX according to the following requirements:
- User student on serverX can log in with a SSH public key to the student account on desktopX.
 - Disable **ssh** login for the root user and password-based SSH authentication on serverX.
10. Your serverX machine has been relocated to the Bahamas. The following changes have to be made on the serverX machine:
- Change the time zone on the serverX machine to Bahamas and verify the time zone has been changed properly.
11. Record the command to display all **systemd** journal entries recorded between 9:05:00 and 9:15:00 in the **/home/student/systemdreview.txt** file.
12. Configure **rsyslogd** by adding a rule to the newly created configuration file **/etc/rsyslog.d/auth-errors.conf** to log all security and authentication messages that get recorded in the authpriv facility with the priority alert and higher to the **/var/log/auth-errors** file as well. Test the newly added log directive with the **logger** command.
13. Create a new static network connection using the settings in the following table. Be sure to replace the X with the correct number for your systems.

- Configure the new connection to be automatically started.
- Other connections should not start automatically.
- Modify the new connection so that it also uses the address 10.0.X.1/24.
- Configure the **hosts** file so that 10.0.X.1 can be referenced as "myhost".
- Set the host name to serverX.example.com.

Parameter	Setting
Connection name	review
IP address	172.25.X.11/24
Gateway address	172.25.X.254
DNS address	172.25.254.254

- Synchronize the **/etc** directory tree on serverX to the **/configbackup** directory on serverX.
- Create an archive named **/root/configuration-backup-server.tar.gz** with the **/configbackup** directory as content.
- To prepare the archived directory tree for comparison with the currently actively used configuration files on serverX, extract the contents of the **/root/configuration-backup-server.tar.gz** archive to the **/tmp/configcompare/** directory on serverX.
- Perform the following tasks on the serverX machine:
 - Use **ssh** to execute the **hostname** command on desktopX as user student. Send the output of the **hostname** command to the **/tmp/scpfile.txt** file on desktopX.
 - Use **scp** to copy the **/tmp/scpfile.txt** file from desktopX to **/home/student/scpfile.txt**.
- Create the file **/etc/yum.repos.d/localupdates.repo** to enable the "Updates" repository found on the content machine. It should access content found at the following URL: http://content.example.com/rhel7.0/x86_64/errata. Do *not* check GPG signatures.
- Configure serverX to adhere to specific software requirements:
 - The **kernel** package should be updated to the latest version.
 - The **xsane-gimp** package should be installed.
 - The **rht-system** package should be installed.
 - For security reasons, serverX should not have the **wvdial** package installed.

FIXME: need to wait for final bits including real available errata packages.
- Generate a disk usage report with the **du** command of the **/usr/share/fonts** directory on serverX and save the result in the **/home/student/dureport.txt** file.

21. Identify and mount a newly added file system by UUID on the **/mnt/datadump** directory on serverX.
22. Create the soft link **/root/mydataspace**, which points to the **/mnt/datadump** directory on serverX.
23. Record the command to find all soft links on serverX that have **data** as part of their name in the **/home/student/find.txt** file.

Solution

In this lab, you will practice and demonstrate your knowledge and skills.

Outcomes:

Complete the following tasks and successfully grade the serverX system with **lab sa1-review grade** as user root on serverX.

Before you begin...

Reset the serverX machine.

Run the **lab sa1-review setup** as user root on serverX.

1. Use Bash commands to complete the following tasks on the serverX machine:

- Display the first 12 lines of the **/usr/bin/clean-binary-files** file and send the output to the **/home/student/headtail.txt** file.
 - Display the last nine lines of the **/usr/bin/clean-binary-files** file and add the output to the **/home/student/headtail.txt** file.
- 1.1. Display the first 12 lines of the **/usr/bin/clean-binary-files** file and send the command output to the **/home/student/headtail.txt** file.

```
[student@serverX ~]$ head -n 12 /usr/bin/clean-binary-files >/home/student/headtail.txt
```

- 1.2. Display the last nine lines of the **/usr/bin/clean-binary-files** file and add the command output to the **/home/student/headtail.txt** file.

```
[student@serverX ~]$ tail -n 9 /usr/bin/clean-binary-files >>/home/student/headtail.txt
```

2. Ten new Linux systems require change documentation files. Carry out the following tasks on serverX to create them:

- Create the empty files with the file name **system_changes-machineY-month_Z.txt** in the **/home/student** directory on the serverX machine as user student. Replace Y with the machine number and replace Z with the months *jan*, *feb*, and *mar*.
- Create the **/home/student/syschanges** directory with the subdirectories **jan**, **feb**, and **mar**.
- Sort all newly created files by month into the corresponding subdirectory.
- Remove all newly created files related to machine 9 and 10, because the hardware has been replaced permanently.

- 2.1. Create a total of 30 files with names **system_changes-machineY-month_Z.txt**. Replace Y with the machine number and replace Z with the months *jan*, *feb*, and *mar*.

```
[student@serverX ~]$ touch ~student/system_changes-machine{1..10}-month_{jan,feb,mar}.txt
```

- 2.2. Create the **/home/student/syschanges** directory with the subdirectories **jan**, **feb**, and **mar**.

```
[student@serverX ~]$ mkdir -p /home/student/syschanges/{jan,feb,mar}
```

- 2.3. Sort all newly created files by month into the corresponding subdirectory.

```
[student@serverX ~]$ mv ~student/system_changes-machine*jan.txt /home/student/syschanges/jan/
[student@serverX ~]$ mv ~student/system_changes-machine*feb.txt /home/student/syschanges/feb/
[student@serverX ~]$ mv ~student/system_changes-machine*mar.txt /home/student/syschanges/mar/
```

- 2.4. Remove all newly created files related to machine 9 and 10.

```
[student@serverX ~]$ rm -f /home/student/syschanges/*/*system_changes-machine{9,10}*.txt
```

3. Use the man pages to research how to turn off the use of colors in the output. Put the relevant option of the **ls** command into the text file **/home/student/lscolor.txt** on serverX.
- 3.1. Look up the relevant option in the **ls(1)** man page to determine how to prevent ls from providing colorful output. What is the correct option?

```
[student@serverX ~]$ man ls
```

ls uses **--color=never** to turn off colors in the command output.

- 3.2. Create the text file **/home/student/lscolor.txt** with the **ls** option to turn off colorful output.

```
[student@serverX ~]$ echo "--color=never" >/home/student/lscolor.txt
```

4. Copy the file **/home/student/vimfile.txt** to **/home/student/longlisting.txt** on serverX. Use the **vim** editor to change the **/home/student/longlisting.txt** file according to the following requirements:

- Remove the file owner column. Do not remove any spaces.
- Remove the **Documents** and **Pictures** rows.
- Save the file when done with editing.

- 4.1. Copy the file **/home/student/vimfile.txt** to **/home/student/longlisting.txt**.

```
[student@serverX ~]$ cp /home/student/vimfile.txt /home/student/longlisting.txt
```

- 4.2. Edit the file using Vim, to take advantage of *visual mode*.

```
[student@serverX ~]$ vim /home/student/longlisting.txt
```

- 4.3. Remove the *owner* column from the file.

Use the arrow keys to position the cursor at the first character of the group owner column. Enter visual mode with **Ctrl-v**. Use the arrow keys to position the cursor at the last character and row of the group owner column. Delete the selection with **x**.

- 4.4. Remove the **Documents** and **Pictures** rows. This time, enter visual mode with an uppercase **V**, which automatically selects full lines.

Use the arrow keys to position the cursor at any character on the **Documents** row. Enter visual mode with an uppercase **V**. The full line is selected, as shown in the screen shot. Delete the selection with **x**. Repeat for the **Pictures** row.

- 4.5. Save the file and exit the editor.

Press the "esc" key and enter ":wq" to write the file and exit **vim**.

5. Change configuration and add new users and a new group according to the following requirements:

- Change the default system settings for newly created users to ensure their passwords are changed at least every 60 days.
- Create a new group named **instructors** with a GID of 30000.
- Create three new users: **gorwell**, **rbradbury**, and **dadams**, with a password of **firstpw**.
- Add the new users to the supplementary group **instructors**. The primary group should remain as the user private group.
- Set the three newly created accounts to expire 60 days from today.
- Change the password policy for the **gorwell** account to require a new password every 10 days.
- Force all three newly created users to change their password on first login.

- 5.1. Change the default system settings for newly created users to ensure their passwords are changed at least every 60 days.

```
[student@serverX ~]$ sudo vim /etc/login.defs
[student@serverX ~]$ cat /etc/login.defs
...Output omitted...
PASS_MAX_DAYS 60
PASS_MIN_DAYS 0
PASS_MIN_LEN 5
PASS_WARN_AGE 7
...Output omitted...
```

- 5.2. Create a new group named **instructors** with a GID of 30000.

```
[student@serverX ~]$ sudo groupadd -g 30000 instructors
[student@serverX ~]$ tail -5 /etc/group
stapdev:x:158:
pesign:x:989:
tcpdump:x:72:
slocate:x:21:
instructors:x:30000:
```

- 5.3. Create three new users: **gorwell**, **rbradbury**, and **dadams**, with a password of **firstpw** and add them to the supplementary group **instructors**. The primary group should remain as the user private group.

```
[student@serverX ~]$ sudo useradd -G instructors gorwell
[student@serverX ~]$ sudo useradd -G instructors rbradbury
[student@serverX ~]$ sudo useradd -G instructors dadams
[student@serverX ~]$ tail -5 /etc/group
slocate:x:21:
instructors:x:30000:gorwell,rbradbury,dadams
gorwell:x:1001:
rbradbury:x:1002:
dadams:x:1003:
[student@serverX ~]$ sudo passwd gorwell
Changing password for user gorwell.
New password: firstpw
BAD PASSWORD: The password is shorter than 8 characters
Retype new password: firstpw
passwd: all authentication tokens updated successfully.
[student@serverX ~]$ sudo passwd rbradbury
[student@serverX ~]$ sudo passwd dadams
```

- 5.4. Determine the date 60 days in the future and set each of the three new user accounts to expire on that date.

```
[student@serverX ~]$ date -d "+60 days"
Mon April  5 11:49:24 EDT 2014
[student@serverX ~]$ sudo chage -E 2014-04-05 gorwell
[student@serverX ~]$ sudo chage -E 2014-04-05 rbradbury
[student@serverX ~]$ sudo chage -E 2014-04-05 dadams
```

- 5.5. Change the password policy for the **gorwell** account to require a new password every 10 days.

```
[student@serverX ~]$ sudo chage -M 10 gorwell
[student@serverX ~]$ chage -l gorwell
Last password change : Feb 04, 2014
Password expires      : Feb 14, 2014
Password inactive     : never
Account expires        : April 05, 2014
Minimum number of days between password change : 0
Maximum number of days between password change : 10
Number of days of warning before password expires : 7
```

- 5.6. Force all three newly created users to change their password on first login.

```
[student@serverX ~]$ sudo chage -d 0 gorwell
```

```
[student@serverX ~]$ sudo chage -d 0 rbradbury  
[student@serverX ~]$ sudo chage -d 0 dadams
```

6. Create the shared directory **/home/instructors** on serverX according to the following requirements:

- The directory is owned by user root and group instructors.
- Set permissions on the **/home/instructors** directory so it has the GID bit set on the directory, the owner and group have full read/write/execute permissions, and other users have read permission to the directory.

- 6.1. Open a terminal window and become root on serverX.

```
[student@serverX ~]$ su -  
Password: redhat  
[root@serverX ~]#
```

- 6.2. Create the **/home/instructors** directory.

```
[root@serverX ~]# mkdir /home/instructors
```

- 6.3. Change group permissions on the **/home/instructors** directory so it belongs to the group instructors.

```
[root@serverX ~]# chown :instructors /home/instructors
```

- 6.4. Set permissions on the **/home/instructors** directory so it is a set GID bit directory (2), the owner (7) and group (7) have full read/write/execute permissions, and other users have read permission (4) to the directory.

```
[root@serverX ~]# chmod 2774 /home/instructors
```

- 6.5. Check that the permissions were set properly.

```
[root@serverX ~]# ls -ld /home/instructors  
drwxrwsr--. 2 root instructors 1024 Dec 9 1:38 /home/instructors
```

7. Determine the process using the most CPU resources on serverX and terminate it.

- 7.1. In a terminal window, run the **top** utility. Size the window as tall as possible. Top sorts all processes by CPU utilization. The cpuhog process is the one with the highest CPU usage.

```
[root@serverX ~]# top  
top - 12:47:46 up 2:02, 3 users, load average: 1.67, 1.25, 0.73  
Tasks: 361 total, 6 running, 355 sleeping, 0 stopped, 0 zombie  
%Cpu(s): 98.5 us, 1.4 sy, 0.0 ni, 0.0 id, 0.0 wa, 0.0 hi, 0.1 si, 0.0 st  
KiB Mem: 2043424 total, 897112 used, 1146312 free, 1740 buffers  
KiB Swap: 4079612 total, 0 used, 4079612 free. 296276 cached Me  
  
 PID USER      PR NI      VIRT      RES      SHR S %CPU %MEM     TIME+ COMMAND  
 4019 root      20  0      4156      76          0 R 57.5  0.0    2:54.15 cpuhog
```

```

2492 student 20 0 1359500 168420 37492 S 16.8 8.2 3:55.58 gnome-shell
1938 root 20 0 189648 35972 7568 R 1.9 1.8 0:29.66 Xorg
2761 student 20 0 620192 19688 12296 S 0.4 1.0 0:04.48 gnome-terminal+
output truncated

```

7.2. Exit the **top** display.

Press **q** to quit.

7.3. Terminate the cphuhog process using the command line. Confirm that the processes no longer display in **top**.

```
[root@serverX ~]# pkill cphuhog
```

8. Stop the currently running cups printing service on serverX. The service should not get automatically started on system boot.

8.1. Stop the **cups** service.

```
[student@serverX ~]$ sudo systemctl stop cups
[student@serverX ~]$ sudo systemctl status cups
```

8.2. Configure the **cups** service so that it does not start at system boot.

```
[student@serverX ~]$ sudo systemctl disable cups
[student@serverX ~]$ sudo systemctl status cups
```

9. Configure the ssh service on serverX according to the following requirements:

- User student on serverX can log in with a SSH public key to the student account on desktopX.
- Disable **ssh** login for the root user and password-based SSH authentication on serverX.

9.1. Generate a SSH public key on serverX as user student.

```
[student@serverX ~]$ ssh-keygen
```

9.2. Install the SSH public key generated previously on serverX to the **student** account on desktopX.

```
[student@serverX ~]$ ssh-copy-id desktopX
/usr/bin/ssh-copy-id: INFO: attempting to log in with the new key(s), to filter
out any that are already installed
/usr/bin/ssh-copy-id: INFO: 1 key(s) remain to be installed -- if you are
prompted now it is to install the new keys
student@desktopX's password: student

Number of key(s) added: 1

Now try logging into the machine, with: "ssh 'student@desktopX'"
and check to make sure that only the key(s) you wanted were added.
```

9.3. Log in, then change to the root account, on the serverX virtual machine.

```
[student@desktopX ~]$ su -
```

9.4. Customize the ssh service on serverX by disabling SSH connections for the user root and only allow key-based login.

Set the necessary configuration file parameters in **/etc/ssh/sshd_config**:

```
PermitRootLogin no  
PasswordAuthentication no
```

9.5. Restart the sshd service on serverX.

```
[root@serverX ~]# systemctl restart sshd
```

9.6. On a different terminal window on desktopX, validate that user root cannot connect to serverX with the **ssh** command. It should fail because we disabled root logins with the ssh service.

```
[student@desktopX ~]$ ssh root@serverX  
Password: redhat  
Permission denied, please try again.  
Password: redhat  
Permission denied, please try again.  
Password: redhat  
Permission denied (publickey,gssapi-keyex,gssapi-with-mic,password).
```

10. Your serverX machine has been relocated to the Bahamas. The following changes have to be made on the serverX machine:

- Change the time zone on the serverX machine to Bahamas and verify the time zone has been changed properly.

10.1. Identify the correct time zone for Bahamas on serverX.

```
[root@serverX ~]# tzselect  
Please identify a location so that time zone rules can be set correctly.  
Please select a continent or ocean.  
1) Africa  
2) Americas  
3) Antarctica  
4) Arctic Ocean  
5) Asia  
6) Atlantic Ocean  
7) Australia  
8) Europe  
9) Indian Ocean  
10) Pacific Ocean  
11) none - I want to specify the time zone using the Posix TZ format.  
#? 2  
Please select a country.  
1) Anguilla          28) Haiti  
2) Antigua & Barbuda    29) Honduras
```

3) Argentina	30) Jamaica
4) Aruba	31) Martinique
5) Bahamas	32) Mexico
6) Barbados	33) Montserrat
7) Belize	34) Nicaragua
8) Bolivia	35) Panama
9) Brazil	36) Paraguay
10) Canada	37) Peru
11) Caribbean Netherlands	38) Puerto Rico
12) Cayman Islands	39) St Barthelemy
13) Chile	40) St Kitts & Nevis
14) Colombia	41) St Lucia
15) Costa Rica	42) St Maarten (Dutch part)
16) Cuba	43) St Martin (French part)
17) Curacao	44) St Pierre & Miquelon
18) Dominica	45) St Vincent
19) Dominican Republic	46) Suriname
20) Ecuador	47) Trinidad & Tobago
21) El Salvador	48) Turks & Caicos Is
22) French Guiana	49) United States
23) Greenland	50) Uruguay
24) Grenada	51) Venezuela
25) Guadeloupe	52) Virgin Islands (UK)
26) Guatemala	53) Virgin Islands (US)
27) Guyana	
#? 5	

The following information has been given:

Bahamas

Therefore TZ='America/Nassau' will be used.
 Local time is now: Fri Mar 7 09:38:50 EST 2014.
 Universal Time is now: Fri Mar 7 14:38:50 UTC 2014.
 Is the above information OK?

- 1) Yes
- 2) No

#? 1

You can make this change permanent for yourself by appending the line
 TZ='America/Nassau'; export TZ
 to the file '.profile' in your home directory; then log out and log in again.

Here is that TZ value again, this time on standard output so that you
 can use the /usr/bin/tzselect command in shell scripts:
 America/Nassau

10.2.Change the time zone to America/Nassau on serverX.

```
[root@serverX ~]# timedatectl set-timezone America/Nassau
```

10.3.Verify the time zone has been properly set on serverX.

```
[root@serverX ~]# timedatectl
  Local time: Wed 2014-04-09 18:21:06 CEST
  Universal time: Wed 2014-04-09 16:21:06 UTC
    RTC time: Wed 2014-04-09 16:21:06
   Timezone: America/Nassau (CEST, +0200)
     NTP enabled: yes
    NTP synchronized: no
      RTC in local TZ: no
```

```
DST active: yes
Last DST change: DST began at
    Sun 2014-03-30 01:59:59 CET
    Sun 2014-03-30 03:00:00 CEST
Next DST change: DST ends (the clock jumps one hour backwards) at
    Sun 2014-10-26 02:59:59 CEST
    Sun 2014-10-26 02:00:00 CET
```

11. Record the command to display all **systemd** journal entries recorded between 9:05:00 and 9:15:00 in the **/home/student/systemdreview.txt** file.

```
[root@serverX ~]# echo "journalctl --since 9:05:00 --until 9:15:00" >/home/student/systemdreview.txt
```

12. Configure **rsyslogd** by adding a rule to the newly created configuration file **/etc/rsyslog.d/auth-errors.conf** to log all security and authentication messages that get recorded in the authpriv facility with the priority alert and higher to the **/var/log/auth-errors** file as well. Test the newly added log directive with the **logger** command.

- 12.1. Add the directive to log **authpriv.alert** syslog messages to the **/var/log/auth-errors** file in the **/etc/rsyslog.d/auth-errors.conf** configuration file.

```
[root@serverX ~]# echo "authpriv.alert /var/log/auth-errors" >/etc/rsyslog.d/auth-errors.conf
```

- 12.2. Restart the **rsyslog** service on serverX.

```
[root@serverX ~]# systemctl restart rsyslog
```

- 12.3. Monitor the newly created log file **/var/log/auth-errors** on serverX for changes in a different terminal window.

```
[root@serverX ~]# tail -f /var/log/auth-errors
```

- 12.4. Use **logger** to create a new log entry to **/var/log/auth-errors** on serverX.

```
[root@serverX ~]# logger -p authpriv.alert "Logging test authpriv.alert"
```

- 12.5. Verify the message sent to syslog with the **logger** command appears in the **/var/log/auth-errors** file on serverX in the terminal running **tail -f /var/log/auth-errors**.

```
[root@serverX ~]# tail -f /var/log/auth-errors
Feb 13 11:21:53 server1 root: Logging test authpriv.alert
```

13. Create a new static network connection using the settings in the following table. Be sure to replace the X with the correct number for your systems.

- Configure the new connection to be automatically started.
- Other connections should not start automatically.
- Modify the new connection so that it also uses the address 10.0.X.1/24.
- Configure the **hosts** file so that 10.0.X.1 can be referenced as "myhost".
- Set the host name to serverX.example.com.

Parameter	Setting
Connection name	review
IP address	172.25.X.11/24
Gateway address	172.25.X.254
DNS address	172.25.254.254

- 13.1. Create a new static network connection using the settings in the table. Be sure to replace the X with the correct number for your systems.

```
[root@serverX ~]# nmcli con add con-name review ifname eth0 type ethernet ip4
172.25.X.11/24 gw4 172.25.X.254
[root@serverX ~]# nmcli con mod "review" ipv4.dns 172.25.254.254
```

- 13.2. Configure the new connection to be autostarted. Other connections should not start automatically.

```
[root@serverX ~]# nmcli con mod "review" connection.autoconnect yes
[root@serverX ~]# nmcli con mod "System eth0" connection.autoconnect no
```

- 13.3. Modify the new connection so that it also uses the address 10.0.X.1/24.

```
[root@serverX ~]# nmcli con mod "review" +ipv4.addresses 10.0.X.1/24
```

Or alternately:

```
[root@serverX ~]# echo "IPADDR1=10.0.X.1" >> /etc/sysconfig/network-scripts/ifcfg-review
[root@serverX ~]# echo "PREFIX1=24" >> /etc/sysconfig/network-scripts/ifcfg-review
```

- 13.4. Configure the **hosts** file so that 10.0.X.1 can be referenced as "myhost".

```
[root@serverX ~]# echo "10.0.X.1 myhost" >> /etc/hosts
```

- 13.5. Set the host name to serverX.example.com.

```
[root@serverX ~]# hostnamectl set-hostname serverX.example.com
```

14. Synchronize the **/etc** directory tree on serverX to the **/configbackup** directory on serverX.

- 14.1. To be able to create the target directory **/configbackup**, switch to the root user account with the **su** command.

```
[student@serverX ~]$ su -  
Password: redhat  
[root@desktopX ~]#
```

- 14.2. Create the target directory for the configuration files on serverX.

```
[root@serverX ~]# mkdir /configbackup
```

- 14.3. Use the **rsync** command to synchronize the **/etc** directory tree on serverX to the **/configsync** directory on serverX. Be aware that only the root user can read all the content in the **/etc** directory on serverX.

```
[root@serverX ~]# rsync -av /etc /configbackup  
...
```

15. Create an archive named **/root/configuration-backup-server.tar.gz** with the **/configbackup** directory as content.

- 15.1. Store the **/configbackup** directory in the **/root/configuration-backup-server.tar.gz** archive.

```
[root@serverX ~]# tar czf /root/configuration-backup-server.tar.gz /configbackup
```

16. To prepare the archived directory tree for comparison with the currently actively used configuration files on serverX, extract the contents of the **/root/configuration-backup-server.tar.gz** archive to the **/tmp/configcompare/** directory on serverX.

- 16.1. Connect to the serverX machine as user root by using **ssh**.

```
[root@desktopX ~]# ssh root@serverX  
Password: redhat  
[root@serverX ~]#
```

- 16.2. Create the target directory **/tmp/configcompare/** where the contents of the **/root/configuration-backup-server.tar.gz** archive will get extracted.

```
[root@serverX ~]# mkdir /tmp/configcompare
```

- 16.3. Change to the target directory **/tmp/configcompare/** on serverX.

```
[root@serverX ~]# cd /tmp/configcompare  
[root@serverX configcompare]#
```

- 16.4.Extract the contents of the `/root/configuration-backup-server.tar.gz` archive to the `/tmp/configcompare/` directory on serverX.

```
[root@serverX configcompare]# tar xzf /root/config-backup-server.tar.gz
```

17. Perform the following tasks on the serverX machine:
- Use `ssh` to execute the `hostname` command on desktopX as user student. Send the output of the `hostname` command to the `/tmp/scpfile.txt` file on desktopX.
 - Use `scp` to copy the `/tmp/scpfile.txt` file from desktopX to `/home/student/scpfile.txt`.
- 17.1. Use `ssh` to execute the `hostname` command on desktopX as user student. Send the output of the `hostname` command to the `/tmp/scpfile.txt` file on desktopX.

```
[root@serverX ~]# ssh student@desktopX 'hostname >/tmp/scpfile.txt'
```

- 17.2. Use `scp` to copy the `/tmp/scpfile.txt` file from desktopX to `/home/student/scpfile.txt` on the serverX machine.

```
[root@serverX ~]# scp root@desktopX:/tmp/scpfile.txt /home/student/
```

18. Create the file `/etc/yum.repos.d/localupdates.repo` to enable the “Updates” repository found on the content machine. It should access content found at the following URL: http://content.example.com/rhel7.0/x86_64/errata. Do not check GPG signatures.

Create the file `/etc/yum.repos.d/localupdates.repo` with the following content:

```
[updates]
name=Red Hat Updates
baseurl=http://content.example.com/rhel7.0/x86_64/errata
enabled=1
gpgcheck=0
```

19. Configure serverX to adhere to specific software requirements:

- The `kernel` package should be updated to the latest version.
- The `xsane-gimp` package should be installed.
- The `rht-system` package should be installed.
- For security reasons, serverX should not have the `wvdial` package installed.

FIXME: need to wait for final bits including real available errata packages.

- 19.1. Update the `kernel` package.

```
yum update kernel
```

- 19.2. Install the `xsane-gimp` package.

```
yum install xsane-gimp
```

19.3. Install the **rht-system** package.

```
yum install rht-system
```

19.4. For security reasons, serverX should not have the **wvdial** package installed.

```
yum remove wvdial
```

20. Generate a disk usage report with the **du** command of the **/usr/share/fonts** directory on serverX and save the result in the **/home/student/dureport.txt** file.

```
[root@serverX ~]# du /usr/share/fonts >/home/student/dureport.txt
```

21. Identify and mount a newly added file system by UUID on the **/mnt/datadump** directory on serverX.

21.1. Identify the newly added file system with the **blkid** command on serverX.

```
[root@serverX ~]# blkid  
/dev/vda1: UUID="46f543fd-78c9-4526-a857-244811be2d88" TYPE="xfs"  
/dev/vdb1: UUID="a84f6842-ec1d-4f6d-b767-b9570f9fc0" TYPE="xfs"
```

21.2. Create the mount point **/mnt/datadump** on serverX.

```
[root@serverX ~]# mkdir /mnt/datadump
```

21.3. Mount the file system by UUID on the **/mnt/datadump** directory of the serverX machine.

```
[root@serverX ~]# mount UUID="a84f6842-ec1d-4f6d-b767-b9570f9fc0" /mnt/  
datadump
```

22. Create the soft link **/root/mydataspace**, which points to the **/mnt/datadump** directory on serverX.

```
[root@serverX ~]# ln -s /mnt/datadump /root/mydataspace
```

23. Record the command to find all soft links on serverX that have **data** as part of their name in the **/home/student/find.txt** file.

```
[root@serverX ~]# echo "find / -type l -name '*data*'" >/home/student/find.txt
```

Summary

Red Hat System Administration I Comprehensive Review

- Review chapters to validate knowledge level.
- Review practice exercises to validate skill level.



redhat®