



Red Hat Training and Certification

Student Workbook (ROLE)

Red Hat Enterprise Linux 8.2 RH124

Red Hat System Administration I

Edition 1



Red Hat System Administration I



Red Hat Enterprise Linux 8.2 RH124
Red Hat System Administration I
Edition 1 20200928
Publication date 20200928

Authors: Fiona Allen, Victor Costea, Snehangshu Karmakar, Marc Kesler,
Ed Parenti, Saumik Paul, Hervé Quatremain, Dallas Spohn
Editor: Steven Bonneville, Ralph Rodriguez, David Sacco, Nicole Muller, Heather
Charles, David O'Brien, Seth Kenlon

Copyright © 2020 Red Hat, Inc.

The contents of this course and all its modules and related materials, including handouts to audience members, are
Copyright © 2020 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 send
email to training@redhat.com or phone toll-free (USA) +1 (866) 626-2994 or +1 (919) 754-3700.

Red Hat, Red Hat Enterprise Linux, the Red Hat 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 the Square O Design, together or apart, are trademarks or registered trademarks
of OpenStack Foundation in the United States and other countries, and are used with the OpenStack Foundation's
permission. Red Hat, Inc. is not affiliated with, endorsed by, or sponsored by the OpenStack Foundation or the
OpenStack community.

All other trademarks are the property of their respective owners.

Contributors: Artur Glogowski, Latha Murthy, Samik Sanyal, Chetan Tiwary, Achyut Madhusudan,
Rudolf Kastl, Rob Locke, Michael Phillips

Document Conventions	ix
Introduction	xi
Red Hat System Administration I	xi
Orientation to the Classroom Environment	xii
Internationalization	xv
1. Getting Started with Red Hat Enterprise Linux	1
What is Linux?	2
Quiz: Getting Started with Red Hat Enterprise Linux	9
Summary	11
2. Accessing the Command Line	13
Accessing the Command Line	14
Quiz: Accessing the Command Line	19
Accessing the Command Line Using the Desktop	23
Guided Exercise: Accessing the Command Line Using the Desktop	28
Executing Commands Using the Bash Shell	30
Quiz: Executing Commands Using the Bash Shell	36
Lab: Accessing the Command Line	40
Summary	46
3. Managing Files From the Command Line	47
Describing Linux File System Hierarchy Concepts	48
Quiz: Describing Linux File System Hierarchy Concepts	51
Specifying Files by Name	55
Quiz: Specifying Files by Name	60
Managing Files Using Command-line Tools	64
Guided Exercise: Managing Files Using Command-line Tools	70
Making Links Between Files	75
Guided Exercise: Making Links Between Files	79
Matching File Names with Shell Expansions	81
Quiz: Matching File Names with Shell Expansions	86
Lab: Managing Files from the Command Line	90
Summary	100
4. Getting Help in Red Hat Enterprise Linux	101
Reading Manual Pages	102
Guided Exercise: Reading Manual Pages	106
Reading Info Documentation	110
Guided Exercise: Reading Info Documentation	113
Lab: Getting Help in Red Hat Enterprise Linux	115
Summary	122
5. Creating, Viewing, and Editing Text Files	123
Redirecting Output to a File or Program	124
Quiz: Redirecting Output to a File or Program	130
Editing Text Files from the Shell Prompt	134
Guided Exercise: Editing Text Files from the Shell Prompt	138
Changing the Shell Environment	140
Guided Exercise: Changing the Shell Environment	146
Lab: Creating, Viewing, and Editing Text Files	149
Summary	157
6. Managing Local Users and Groups	159
Describing User and Group Concepts	160
Quiz: Describing User and Group Concepts	163
Gaining Superuser Access	167

Guided Exercise: Gaining Superuser Access	172
Managing Local User Accounts	177
Guided Exercise: Managing Local User Accounts	181
Managing Local Group Accounts	184
Guided Exercise: Managing Local Group Accounts	187
Managing User Passwords	190
Guided Exercise: Managing User Passwords	194
Lab: Managing Local Users and Groups	198
Summary	203
7. Controlling Access to Files	205
Interpreting Linux File System Permissions	206
Quiz: Interpreting Linux File System Permissions	210
Managing File System Permissions from the Command Line	214
Guided Exercise: Managing File System Permissions from the Command Line	218
Managing Default Permissions and File Access	222
Guided Exercise: Managing Default Permissions and File Access	226
Lab: Controlling Access to Files	231
Summary	237
8. Monitoring and Managing Linux Processes	239
Listing Processes	240
Quiz: Listing Processes	245
Controlling Jobs	247
Guided Exercise: Controlling Jobs	250
Killing Processes	255
Guided Exercise: Killing Processes	261
Monitoring Process Activity	266
Guided Exercise: Monitoring Process Activity	270
Lab: Monitoring and Managing Linux Processes	275
Summary	286
9. Controlling Services and Daemons	287
Identifying Automatically Started System Processes	288
Guided Exercise: Identifying Automatically Started System Processes	293
Controlling System Services	297
Guided Exercise: Controlling System Services	301
Lab: Controlling Services and Daemons	305
Summary	309
10. Configuring and Securing SSH	311
Accessing the Remote Command Line with SSH	312
Guided Exercise: Accessing the Remote Command Line	316
Configuring SSH Key-based Authentication	320
Guided Exercise: Configuring SSH Key-based Authentication	324
Customizing OpenSSH Service Configuration	330
Guided Exercise: Customizing OpenSSH Service Configuration	332
Lab: Configuring and Securing SSH	338
Summary	345
11. Analyzing and Storing Logs	347
Describing System Log Architecture	348
Quiz: Describing System Log Architecture	350
Reviewing Syslog Files	354
Guided Exercise: Reviewing Syslog Files	358
Reviewing System Journal Entries	360
Guided Exercise: Reviewing System Journal Entries	365

Preserving the System Journal	369
Guided Exercise: Preserving the System Journal	372
Maintaining Accurate Time	374
Guided Exercise: Maintaining Accurate Time	378
Lab: Analyzing and Storing Logs	382
Summary	387
12. Managing Networking	389
Describing Networking Concepts	390
Quiz: Describing Networking Concepts	402
Validating Network Configuration	406
Guided Exercise: Validating Network Configuration	412
Configuring Networking from the Command Line	415
Guided Exercise: Configuring Networking from the Command Line	421
Editing Network Configuration Files	427
Guided Exercise: Editing Network Configuration Files	430
Configuring Host Names and Name Resolution	435
Guided Exercise: Configuring Host Names and Name Resolution	438
Lab: Managing Networking	442
Summary	447
13. Archiving and Transferring Files	449
Managing Compressed tar Archives	450
Guided Exercise: Managing Compressed Tar Archives	457
Transferring Files Between Systems Securely	460
Guided Exercise: Transferring Files Between Systems Securely	462
Synchronizing Files Between Systems Securely	464
Guided Exercise: Synchronizing Files Between Systems Securely	467
Lab: Archiving and Transferring Files	469
Summary	475
14. Installing and Updating Software Packages	477
Registering Systems for Red Hat Support	478
Quiz: Registering Systems for Red Hat Support	482
Explaining and Investigating RPM Software Packages	484
Guided Exercise: Explaining and Investigating RPM Software Packages	490
Installing and Updating Software Packages with Yum	493
Guided Exercise: Installing and Updating Software Packages with Yum	500
Enabling Yum Software Repositories	505
Guided Exercise: Enabling Yum Software Repositories	508
Managing Package Module Streams	512
Guided Exercise: Managing Package Module Streams	519
Lab: Installing and Updating Software Packages	524
Summary	531
15. Accessing Linux File Systems	533
Identifying File Systems and Devices	534
Quiz: Identifying File Systems and Devices	538
Mounting and Unmounting File Systems	540
Guided Exercise: Mounting and Unmounting File Systems	544
Locating Files on the System	547
Guided Exercise: Locating Files on the System	554
Lab: Accessing Linux File Systems	557
Summary	562
16. Analyzing Servers and Getting Support	563
Analyzing and Managing Remote Servers	564

Guided Exercise: Analyzing and Managing Remote Servers	580
Getting Help From Red Hat Customer Portal	584
Guided Exercise: Getting Help from Red Hat Customer Portal	593
Detecting and Resolving Issues with Red Hat Insights	595
Quiz: Detecting and Resolving Issues with Red Hat Insights	604
Summary	606

17. Comprehensive Review **607**

Comprehensive Review	608
Lab: Managing Files from the Command Line	612
Lab: Managing Users and Groups, Permissions and Processes	620
Lab: Configuring and Managing a Server	627
Lab: Managing Networks	635
Lab: Mounting Filesystems and Finding Files	642

Document Conventions



References

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



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.

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

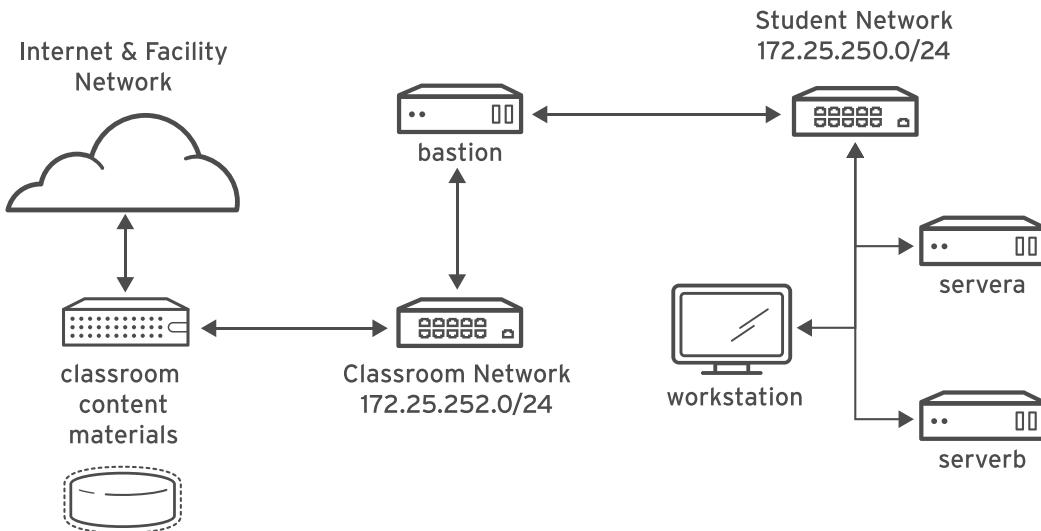


Figure 0.1: Classroom environment

In this course, the main computer system used for hands-on learning activities is **workstation**. Two other machines are also used by students for these activities: **servera**, and **serverb**. All three of these systems are in the **lab.example.com** DNS domain.

All student computer systems have a standard user account, **student**, which has the password **student**. The **root** password on all student systems is **redhat**.

Classroom Machines

Machine name	IP addresses	Role
bastion.lab.example.com	172.25.250.254	Gateway system to connect student private network to classroom server (must always be running)
workstation.lab.example.com	172.25.250.9	Graphical workstation used for system administration
servera.lab.example.com	172.25.250.10	First server
serverb.lab.example.com	172.25.250.11	Second server

The primary function of **bastion** is that it acts as a router between the network that connects the student machines and the classroom network. If **bastion** is down, other student machines will only be able to access systems on the individual student network.

Introduction

Several systems in the classroom provide supporting services. Two servers, **content.example.com** and **materials.example.com**, are sources for software and lab materials used in hands-on activities. Information on how to use these servers is provided in the instructions for those activities. These are provided by the **classroom.example.com** virtual machine. Both **classroom** and **bastion** should always be running for proper use of the lab environment.



Note

When logging on to **servera** or **serverb** you might see a message concerning the activation of **cockpit**. The message can be ignored.

```
[student@workstation ~]$ ssh student@serverb
Warning: Permanently added 'serverb,172.25.250.11' (ECDSA) to the list of
known hosts.
Activate the web console with: systemctl enable --now cockpit.socket

[student@serverb ~]$
```

Controlling Your Systems

You are assigned remote computers in a Red Hat Online Learning classroom. They are accessed through a web application hosted at rol.redhat.com [<http://rol.redhat.com>]. You should log in to this site using your Red Hat Customer Portal user credentials.

Controlling the Virtual Machines

The virtual machines in your classroom environment are controlled through a web page. The state of each virtual machine in the classroom is displayed on the page under the **Online Lab** tab.

Machine States

Virtual Machine State	Description
STARTING	The virtual machine is in the process of booting.
STARTED	The virtual machine is running and available (or, when booting, soon will be).
STOPPING	The virtual machine is in the process of shutting down.
STOPPED	The virtual machine is completely shut down. Upon starting, the virtual machine boots into the same state as when it was shut down (the disk will have been preserved).
PUBLISHING	The initial creation of the virtual machine is being performed.
WAITING_TO_START	The virtual machine is waiting for other virtual machines to start.

Depending on the state of a machine, a selection of the following actions is available.

Classroom/Machine Actions

Button or Action	Description
PROVISION LAB	Create the ROL classroom. Creates all of the virtual machines needed for the classroom and starts them. Can take several minutes to complete.
DELETE LAB	Delete the ROL classroom. Destroys all virtual machines in the classroom. Caution: Any work generated on the disks is lost.
START LAB	Start all virtual machines in the classroom.
SHUTDOWN LAB	Stop all virtual machines in the classroom.
OPEN CONSOLE	Open a new tab in the browser and connect to the console of the virtual machine. You can log in directly to the virtual machine and run commands. In most cases, you should log in to the workstation virtual machine and use ssh to connect to the other virtual machines.
ACTION → Start	Start (power on) the virtual machine.
ACTION → Shutdown	Gracefully shut down the virtual machine, preserving the contents of its disk.
ACTION → Power Off	Forcefully shut down the virtual machine, preserving the contents of its disk. This is equivalent to removing the power from a physical machine.
ACTION → Reset	Forcefully shut down the virtual machine and reset the disk to its initial state. Caution: Any work generated on the disk is lost.

At the start of an exercise, if instructed to reset a single virtual machine node, click **ACTION → Reset** for only the specific virtual machine.

At the start of an exercise, if instructed to reset all virtual machines, click **ACTION → Reset**

If you want to return the classroom environment to its original state at the start of the course, you can click **DELETE LAB** to remove the entire classroom environment. After the lab has been deleted, you can click **PROVISION LAB** to provision a new set of classroom systems.



Warning

The **DELETE LAB** operation cannot be undone. Any work you have completed in the classroom environment up to that point will be lost.

The Autostop Timer

The Red Hat Online Learning enrollment entitles you to a certain amount of computer time. To help conserve allotted computer time, the ROL classroom has an associated countdown timer, which shuts down the classroom environment when the timer expires.

To adjust the timer, click **MODIFY** to display the **New Autostop Time** dialog box. Set the number of hours until the classroom should automatically stop. Note that there is a maximum time of ten hours. Click **ADJUST TIME** to apply this change to the timer settings.

Internationalization

Per-user Language Selection

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

Language Settings

In the GNOME desktop environment, the user might 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.

You can start this application in two ways. You can run the command **gnome-control-center region** from a terminal window, or on the top bar, from the system menu in the right corner, select the settings button (which has a crossed screwdriver and wrench for an icon) from the bottom left of the menu.

In the window that opens, select Region & Language. Click the **Language** box and select the preferred language from the list that appears. This also updates the **Formats** setting to the default for that language. The next time you log in, these changes will take full effect.

These settings affect the GNOME desktop environment and any applications such as **gnome-terminal** that are started inside it. However, by default they do not apply to that account if accessed through an **ssh** login from a remote system or a text-based login on a virtual console (such as **tty5**).



Note

You can make your shell environment use the same **LANG** setting as your graphical environment, even when you log in through a text-based virtual console or over **ssh**. One way to do this is to place code similar to the following in your **~/.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=$(grep 'Language=' /var/lib/AccountsService/users/${USER} \
| sed 's/Language=//')
if [ "$i" != "" ]; then
    export LANG=$i
fi
```

Japanese, Korean, Chinese, and other languages with a non-Latin character set might not display properly on text-based virtual 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 25 17:55:01 CET 2019
```

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

Input Method Settings

GNOME 3 in Red Hat Enterprise Linux 7 or later 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 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.

When 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 also find 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 you know the character's Unicode code point. Type **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, the **root** user can change the system-wide locale settings with the **localectl** command. If **localectl** is run with no arguments, it displays the current system-wide locale settings.

To set the system-wide default language, run the command **localectl set-locale** **LANG=locale**, where *locale* is the appropriate value for the **LANG** environment variable 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 by clicking the **Login Screen** button at the upper-right corner of the window. Changing the **Language** of the graphical login screen will also adjust the system-wide default language setting stored in the **/etc/locale.conf** configuration file.



Important

Text-based virtual consoles such as **tty4** are more limited in the fonts they can display than terminals in a virtual console running a graphical environment, or pseudoterminals for **ssh** sessions. For example, Japanese, Korean, and Chinese characters may not display as expected on a text-based virtual console. For this reason, you should consider using English or another language with a Latin character set for the system-wide default.

Likewise, text-based virtual 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 text-based virtual consoles and the graphical environment. See the **localectl(1)** and **vconsole.conf(5)** man pages for more information.

Language Packs

Special RPM packages called *langpacks* install language packages that add support for specific languages. These langpacks use dependencies to automatically install additional RPM packages containing localizations, dictionaries, and translations for other software packages on your system.

To list the langpacks that are installed and that may be installed, use **yum list langpacks-***:

```
[root@host ~]# yum list langpacks-*
Updating Subscription Management repositories.
Updating Subscription Management repositories.
Installed Packages
langpacks-en.noarch      1.0-12.el8        @AppStream
Available Packages
langpacks-af.noarch       1.0-12.el8        rhel-8-for-x86_64-appstream-rpms
langpacks-am.noarch       1.0-12.el8        rhel-8-for-x86_64-appstream-rpms
langpacks-ar.noarch       1.0-12.el8        rhel-8-for-x86_64-appstream-rpms
langpacks-as.noarch       1.0-12.el8        rhel-8-for-x86_64-appstream-rpms
langpacks-ast.noarch      1.0-12.el8        rhel-8-for-x86_64-appstream-rpms
...output omitted...
```

To add language support, install the appropriate langpacks package. For example, the following command adds support for French:

```
[root@host ~]# yum install langpacks-fr
```

Introduction

Use **yum repoquery --whatsonplements** to determine what RPM packages may be installed by a langpack:

```
[root@host ~]# yum repoquery --whatsonplements langpacks-fr
Updating Subscription Management repositories.
Updating Subscription Management repositories.
Last metadata expiration check: 0:01:33 ago on Wed 06 Feb 2019 10:47:24 AM CST.
glibc-langpack-fr-0:2.28-18.el8.x86_64
gnome-getting-started-docs-fr-0:3.28.2-1.el8.noarch
 hunspell-fr-0:6.2-1.el8.noarch
 hyphen-fr-0:3.0-1.el8.noarch
 libreoffice-langpack-fr-1:6.0.6.1-9.el8.x86_64
 man-pages-fr-0:3.70-16.el8.noarch
 mythes-fr-0:2.3-10.el8.noarch
```

**Important**

Langpacks packages use RPM *weak dependencies* in order to install supplementary packages only when the core package that needs it is also installed.

For example, when installing *langpacks-fr* as shown in the preceding examples, the *mythes-fr* package will only be installed if the *mythes* thesaurus is also installed on the system.

If *mythes* is subsequently installed on that system, the *mythes-fr* package will also automatically be installed due to the weak dependency from the already installed *langpacks-fr* package.

**References**

locale(7), **localectl(1)**, **locale.conf(5)**, **vconsole.conf(5)**, **unicode(7)**, and **utf-8(7)** 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

**Note**

This table might not reflect all langpacks available on your system. Use **yum info langpacks-SUFFIX** to get more information about any particular langpacks package.

Language Codes

Language	Langpacks Suffix	\$LANG value
English (US)	en	en_US.utf8

Language	Langpacks Suffix	\$LANG value
Assamese	as	as_IN.utf8
Bengali	bn	bn_IN.utf8
Chinese (Simplified)	zh_CN	zh_CN.utf8
Chinese (Traditional)	zh_TW	zh_TW.utf8
French	fr	fr_FR.utf8
German	de	de_DE.utf8
Gujarati	gu	gu_IN.utf8
Hindi	hi	hi_IN.utf8
Italian	it	it_IT.utf8
Japanese	ja	ja_JP.utf8
Kannada	kn	kn_IN.utf8
Korean	ko	ko_KR.utf8
Malayalam	ml	ml_IN.utf8
Marathi	mr	mr_IN.utf8
Odia	or	or_IN.utf8
Portuguese (Brazilian)	pt_BR	pt_BR.utf8
Punjabi	pa	pa_IN.utf8
Russian	ru	ru_RU.utf8
Spanish	es	es_ES.utf8
Tamil	ta	ta_IN.utf8
Telugu	te	te_IN.utf8

Chapter 1

Getting Started with Red Hat Enterprise Linux

Goal

Describe and define open source, Linux, Linux distributions, and Red Hat Enterprise Linux.

Objectives

- Define and explain the purpose of Linux, open source, Linux distributions, and Red Hat Enterprise Linux.

Sections

- What is Linux?

Quiz

Getting Started with Red Hat Enterprise Linux

What is Linux?

Objectives

After completing this section, you should be able to define and explain the purpose of Linux, open source, Linux distributions, and Red Hat Enterprise Linux.

Why Should You Learn about Linux?

Linux is a critical technology for IT professionals to understand.

Linux is in widespread use, and if you use the internet at all, you are probably already interacting with Linux systems in your daily life. Perhaps the most obvious way in which you interact with Linux systems is through browsing the World Wide Web and using e-commerce sites to buy and sell products.

However, Linux is in use for much more than that. Linux manages point-of-sale systems and the world's stock markets, and also powers smart TVs and in-flight entertainment systems. It powers most of the top 500 supercomputers in the world. Linux provides the foundational technologies powering the cloud revolution and the tools used to build the next generation of container-based microservices applications, software-based storage technologies, and big data solutions.

In the modern data center, Linux and Microsoft Windows are the major players, and Linux is a growing segment in that space. Some of the many reasons to learn Linux include:

- A Windows user needs to interoperate with Linux.
- In application development, Linux hosts the application or its runtime.
- In cloud computing, the cloud instances in the private or public cloud environment use Linux as the operating system.
- With mobile applications or the Internet of Things (IoT), the chances are high that the operating system of your device uses Linux.
- If you are looking for new opportunities in IT, Linux skills are in high demand.

What Makes Linux Great?

There are many different answers to the question "What makes Linux great?", however, three of them are:

- Linux is *open source* software.

Being open source does not just mean that you can see how the system works. You can also experiment with changes and share them freely for others to use. The open source model means that improvements are easier to make, enabling faster innovation.

- Linux provides easy access to a powerful and scriptable *command-line interface (CLI)*.

Linux was built around the basic design philosophy that users can perform all administration tasks from the CLI. It enables easier automation, deployment, and provisioning, and simplifies both local and remote system administration. Unlike other operating systems, these capabilities

have been built in from the beginning, and the assumption has always been to enable these important capabilities.

- Linux is a modular operating system that allows you to easily replace or remove components.

Components of the system can be upgraded and updated as needed. A Linux system can be a general-purpose development workstation or an extremely stripped-down software appliance.

What is Open Source Software?

Open source software is software with source code that anyone can use, study, modify, and share.

Source code is the set of human-readable instructions that are used to make a program. It may be interpreted as a script or compiled into a binary executable which the computer runs directly. Upon creating source code, it gets copyrighted, and the copyright holder controls the terms under which the software can be copied, adapted, and distributed. Users can use this software under a software license.

Some software has source code that only the person, team, or organization that created it can see, or change, or distribute. This software is sometimes called "proprietary" or "closed source" software. Typically the license only allows the end user to run the program, and provides no access, or tightly limited access, to the source.

Open source software is different. When the copyright holder provides software under an open source license, they grant the user the right to run the program and *also* to view, modify, compile, and redistribute the source royalty-free to others.

Open source promotes collaboration, sharing, transparency, and rapid innovation because it encourages people beyond the original developers to make modifications and improvements to the software and to share it with others.

Just because the software is open source does not mean it is somehow not able to be used or provided commercially. Open source is a critical part of many organizations' commercial operations. Some open source licenses allow code to be reused in closed source products. One can sell open source code, but the terms of true open source licenses generally allow the customer to redistribute the source code. Most commonly, vendors such as Red Hat provide commercial help with deploying, supporting, and extending solutions based on open source products.

Open source has many benefits for the user:

- *Control*: See what the code does and change it to improve it.
- *Training*: Learn from real-world code and develop more useful applications.
- *Security*: Inspect sensitive code, fix with or without the original developers' help.
- *Stability*: Code can survive the loss of the original developer or distributor.

The bottom line is that open source allows the creation of better software with a higher return on investment by collaboration.

Types of Open Source Licenses

There is more than one way to provide open source software. The terms of the software license control how the source can be combined with other code or reused, and hundreds of different open source licenses exist. However, to be open source, licenses must allow users to freely use, view, change, compile, and distribute the code.

There are two broad classes of open source license that are particularly important:

- *Copyleft* licenses that are designed to encourage keeping code open source.
- *Permissive* licenses that are designed to maximize code reusability.

Copyleft, or "share-alike" licenses, require that anyone who distributes the source code, with or without changes, must also pass along the freedom for others to also copy, change, and distribute the code. The basic advantage of these licenses is that they help to keep existing code, and improvements to that code, open and add to the amount of open source code available. Common copyleft licenses include the *GNU General Public License (GPL)* and the *Lesser GNU Public License (LGPL)*.

Permissive licenses are intended to maximize the reusability of source code. Users can use the source for any purpose as long as the copyright and license statements are preserved, including reusing that code under more restrictive or even proprietary licenses. This makes it very easy for this code to be reused, but at the risk of encouraging proprietary-only enhancements. Several commonly used permissive open source licenses include the *MIT/X11 license*, the *Simplified BSD license*, and the *Apache Software License 2.0*.

Who Develops Open Source Software?

It is a misconception to think that open source is developed solely by an "army of volunteers" or even an army of individuals plus Red Hat. Open source development today is overwhelmingly professional. Many developers are paid by their organizations to work with open source projects to construct and contribute the enhancements they and their customers need.

Volunteers and the academic community play a significant role and can make vital contributions, especially in new technology areas. The combination of formal and informal development provides a highly dynamic and productive environment.

Who is Red Hat?

Red Hat is the world's leading provider of open source software solutions, using a community-powered approach to reliable and high-performance cloud, Linux, middleware, storage, and virtualization technologies. Red Hat's mission is to be the catalyst in communities of customers, contributors, and partners creating better technology the open source way.

Red Hat's role is to help customers connect with the open source community and their partners to effectively use open source software solutions. Red Hat actively participates in and supports the open source community and many years of experience have convinced the company of the importance of open source to the future of the IT industry.

Red Hat is most well-known for their participation in the Linux community and the Red Hat Enterprise Linux distribution. However, Red Hat is also very active in other open source communities, including middleware projects centered on the JBoss developer community, virtualization solutions, cloud technologies such as OpenStack and OpenShift, and the Ceph and Gluster software-based storage projects, among others.

What is a Linux distribution?

A *Linux distribution* is an installable operating system constructed from a Linux kernel and supporting user programs and libraries. A complete Linux operating system is not developed by a single organization, but by a collection of independent open source development communities working with individual software components. A distribution provides an easy way for users to install and manage a working Linux system.

In 1991, a young computer science student named Linus Torvalds developed a Unix-like kernel he named *Linux*, licensed as open source software under the GPL. The kernel is the core component

of the operating system, which manages hardware, memory, and the scheduling of running programs. This Linux kernel could then be supplemented with other open source software, such as utilities and programs from the GNU Project, the graphical interface from MIT's *X Window System*, and many other open source components, such as the Sendmail mail server or the Apache HTTP web server, in order to build a complete open source Unix-like operating system.

However, one of the challenges for Linux users was to assemble all these pieces from many different sources. Very early in its history, Linux developers began working to provide a distribution of prebuilt and tested tools that users could download and use to set up their Linux systems quickly.

Many different Linux distributions exist, with differing goals and criteria for selecting and supporting the software provided by their distribution. However, distributions generally have many common characteristics:

- Distributions consist of a Linux kernel and supporting user space programs.
- Distributions can be small and single-purpose or include thousands of open source programs.
- Distributions must provide a means of installing and updating the distribution and its components.
- The provider of the distribution must support that software, and ideally, be participating directly in the community developing that software.

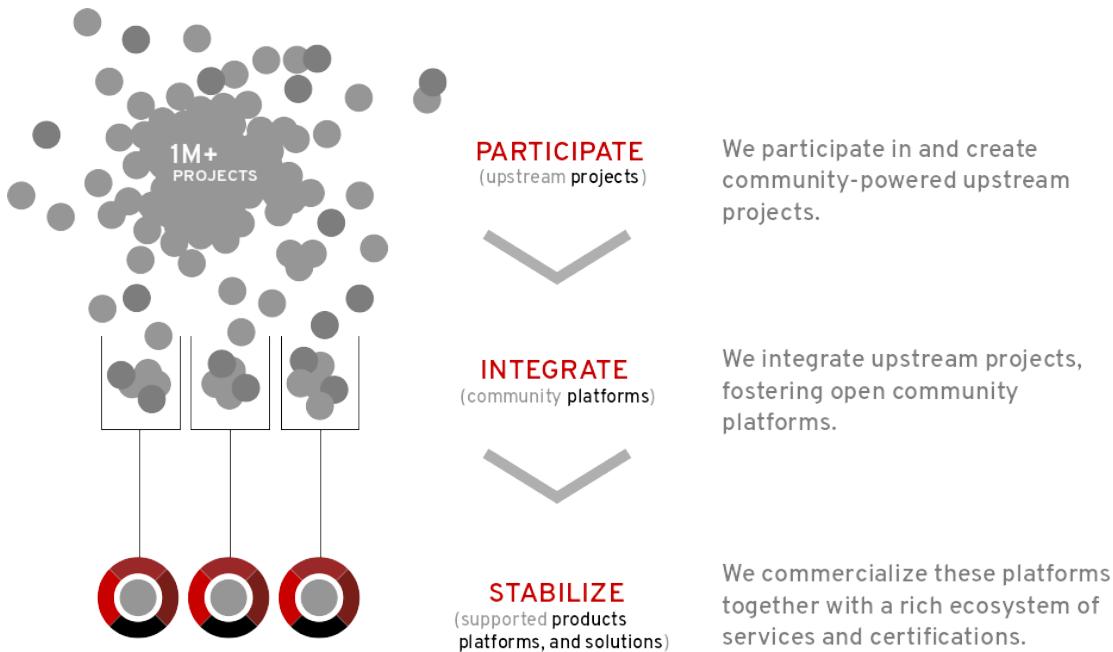
Red Hat Enterprise Linux is Red Hat's commercialized Linux distribution.

Red Hat Enterprise Linux

Development of Red Hat Enterprise Linux

Red Hat develops and integrates open source software into RHEL through a multistage process.

- Red Hat participates in supporting individual open source projects. It contributes code, developer time, resources, and other support, often collaborating with developers from other Linux distributions. It helps to improve the general quality of software for everyone.
- Red Hat sponsors and integrates open source projects into a community-driven Linux distribution, *Fedora*. *Fedora* provides a free working environment that can serve as a development lab and proving ground for features that are incorporated into their commercialized products.
- Red Hat stabilizes the software to ensure that it is ready for long term support and standardization, and integrates it into their enterprise-ready distribution, RHEL.



Fedora

Fedora is a community project that produces and releases a complete, free, Linux-based operating system. Red Hat sponsors the community and works with community representatives to integrate the latest upstream software into a fast-moving and secure distribution. The Fedora project contributes everything back to the free and open source world, and anyone can participate.

However, Fedora focuses on innovation and excellence, not long-term stability. New major updates happen every six months, and they can bring significant changes. Fedora only supports releases for about a year (two major updates), which makes it less suitable for enterprise use.

Red Hat Enterprise Linux

Red Hat Enterprise Linux (RHEL) is Red Hat's enterprise-ready, commercially-supported Linux distribution. It is the leading platform for open source computing, not just a collection of mature open source projects. RHEL is extensively tested, with a large supporting ecosystem of partners, hardware and software certifications, consulting services, training, and multiyear support and maintenance guarantees.

Red Hat bases its major releases of RHEL on Fedora. However, after that Red Hat can pick and choose which packages to include, make further enhancements (contributed back to the upstream projects and Fedora), and make configuration decisions that serve the needs of customers. Red Hat helps vendors and customers engage with the open source community, and to work with upstream development to develop solutions and fix issues.

Red Hat Enterprise Linux uses a subscription-based distribution model. Because this is open source software, this is not a license fee. Instead, it pays for support, maintenance, updates, security patches, access to the Knowledgebase on the Red Hat Customer Portal (<http://access.redhat.com/>), certifications, and so on. The customer is paying for long-term support and expertise, commitment, and assistance when they need it.

When major updates become available, customers can move to them at their convenience without paying more. It simplifies the management of both the economic and practical aspects of system updates.

CentOS

CentOS is a community-driven Linux distribution derived from much of the open source Red Hat Enterprise Linux codebase and other sources. It is free of charge, is easy to install, and is staffed and supported by an active user community of volunteers that operates independently of Red Hat.

The following table lists some key differences between CentOS and Red Hat Enterprise Linux.

CentOS	Red Hat Enterprise Linux
Self-support only.	Several support levels are available, including Standard support during business hours, Premium 24x7 support for critical issues, and entry-level support subscriptions. Different SLA levels can be mixed and matched across an environment.
Errata production starts when an official RHEL errata release is available.	Rapid issue response by in-house developers, hot fixes may be available before the official RHEL errata release.
Package updates are provided for the most recent minor release until the end of RHEL's Maintenance Support 2 phase.	Updates are available for older minor releases under the Extended Update Support (EUS) program, and for years beyond the end of Maintenance Support 2 through the Extended Lifecycle Support (ELS) program.
Generally not certified by software vendors like SAS, SAP, and Oracle as a supported platform.	Thousands of certified applications from hundreds of ISVs.
Help and documentation resources available through forums, mailing lists, chat, the CentOS Project web site and wiki, and other community sources.	Documentation, reference architectures, case studies, and Knowledgebase articles available through Red Hat Customer Portal. Access to Red Hat Customer Portal Labs, a set of tools you can use to improve performance, identify security problems, or assist with any issues. Optional proactive system analysis with Red Hat Insights, a SaaS-based tool to provide real-time assessment of risks related to performance, availability, stability, and security.

Trying out Red Hat Enterprise Linux

There are many different ways to try Red Hat Enterprise Linux. One way is to download an evaluation copy from the website at <https://access.redhat.com/products/red-hat-enterprise-linux/evaluation>. That page includes links to supplementary information.

Red Hat also makes available free subscriptions to a number of products for development purposes through the Red Hat Developer Program at <https://developer.redhat.com>. These subscriptions allow developers to develop quickly, prototype, test, and demonstrate their software to deploy on the same enterprise products.

Another approach is to deploy an instance of Red Hat Enterprise Linux made available through a cloud provider. For example, Red Hat has official AMIs available for Red Hat Enterprise Linux in the Amazon AWS Marketplace.

For more information, visit the Red Hat Enterprise Linux "Get Started" page, referenced at the end of this section.



References

Get Started with Red Hat Enterprise Linux

<https://access.redhat.com/products/red-hat-enterprise-linux#getstarted>

The Open Source Way

<https://opensource.com/open-source-way>

► Quiz

Getting Started with Red Hat Enterprise Linux

Choose the correct answers to the following questions:

► 1. Which two of the following are benefits of open source software for the user? (Choose two.)

- a. Code can survive the loss of the original developer or distributor.
- b. Sensitive portions of code are protected and only available to the original developer.
- c. You can learn from real-world code and develop more effective applications.
- d. Code remains open as long as it is in a public repository but the license may change when included with closed source software.

► 2. Which two of the following are ways in which Red Hat develops their products for the future and interacts with the community? (Choose two.)

- a. Sponsor and integrate open source projects into the community-driven Fedora project.
- b. Develop specific integration tools only available in Red Hat distributions.
- c. Participate in upstream projects.
- d. Repackage and re-license community products.

► 3. Which two statements describe the benefits of Linux? (Choose two.)

- a. Linux is developed entirely by volunteers making it a low cost operating system.
- b. Linux is modular and can be configured as a full graphical desktop or a small appliance.
- c. Linux is locked in a known state for a minimum of one year for each release, making it easier to develop custom software.
- d. Linux includes a powerful and scriptable command-line interface, enabling easier automation and provisioning.

► Solution

Getting Started with Red Hat Enterprise Linux

Choose the correct answers to the following questions:

► 1. Which two of the following are benefits of open source software for the user? (Choose two.)

- a. Code can survive the loss of the original developer or distributor.
- b. Sensitive portions of code are protected and only available to the original developer.
- c. You can learn from real-world code and develop more effective applications.
- d. Code remains open as long as it is in a public repository but the license may change when included with closed source software.

► 2. Which two of the following are ways in which Red Hat develops their products for the future and interacts with the community? (Choose two.)

- a. Sponsor and integrate open source projects into the community-driven Fedora project.
- b. Develop specific integration tools only available in Red Hat distributions.
- c. Participate in upstream projects.
- d. Repackage and re-license community products.

► 3. Which two statements describe the benefits of Linux? (Choose two.)

- a. Linux is developed entirely by volunteers making it a low cost operating system.
- b. Linux is modular and can be configured as a full graphical desktop or a small appliance.
- c. Linux is locked in a known state for a minimum of one year for each release, making it easier to develop custom software.
- d. Linux includes a powerful and scriptable command-line interface, enabling easier automation and provisioning.

Summary

In this chapter, you learned:

- Open source software is software with source code that anyone can freely use, study, modify, and share.
- A Linux distribution is an installable operating system constructed from a Linux kernel and supporting user programs and libraries.
- Red Hat participates in supporting and contributing code to open source projects, sponsors and integrates project software into community-driven distributions, and stabilizes the software to offer it as supported enterprise-ready products.
- Red Hat Enterprise Linux is Red Hat's open source, enterprise-ready, commercially-supported Linux distribution.

Chapter 2

Accessing the Command Line

Goal

Log in to a Linux system and run simple commands using the shell.

Objectives

- Log in to a Linux system on a local text console and run simple commands using the shell.
- Log in to a Linux system using the GNOME 3 desktop environment and run commands from a shell prompt in a terminal program.
- Save time by using tab completion, command history, and command editing shortcuts to run commands in the Bash shell.

Sections

- Accessing the Command Line (and Quiz)
- Accessing the Command Line Using the Desktop (and Guided Exercise)
- Executing Commands Using the Bash Shell (and Quiz)

Lab

Accessing the Command Line

Accessing the Command Line

Objectives

After completing this section, you should be able to log in to a Linux system and run simple commands using the shell.

Introduction to 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*. Various options for the shell program have been developed over the years, and different users can be configured to use different shells. Most users, however, stick with the current default.

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, as shown below.

```
[user@host ~]$
```

The \$ character is replaced by a # character 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 which can affect the whole system. The superuser shell prompt is shown below.

```
[root@host ~]#
```

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 and later. Administrators using the Apple Mac who use the Terminal utility may be pleased to note that **bash** is the default shell in macOS.

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 upon.

For example, the command `usermod -L user01` has a command (`usermod`), an option (`-L`), and an argument (`user01`). The effect of this command is to lock the password of the `user01` user account.

Logging in to a Local Computer

To run the shell, you need to log in to the computer on a *terminal*. A terminal is a text-based interface used to enter commands into and print output from a computer system. There are several ways to do this.

The computer might have a hardware keyboard and display for input and output directly connected to it. This is the Linux machine's *physical console*. The physical console supports multiple *virtual consoles*, which can run separate terminals. Each virtual console supports an independent login session. You can switch between them by pressing **Ctrl+Alt** and a function key (**F1** through **F6**) at the same time. Most of these virtual consoles run a terminal providing a text login prompt, and if you enter your username and password correctly, you will log in and get a shell prompt.

The computer might provide a graphical login prompt on one of the virtual consoles. You can use this to log in to a *graphical environment*. The graphical environment also runs on a virtual console. To get a shell prompt you must start a terminal program in the graphical environment. The shell prompt is provided in an application window of your graphical terminal program.



Note

Many system administrators choose not to run a graphical environment on their servers. This allows resources which would be used by the graphical environment to be used by the server's services instead.

In Red Hat Enterprise Linux 8, if the graphical environment is available, the login screen will run on the first virtual console, called **tty1**. Five additional text login prompts are available on virtual consoles two through six.

If you log in using the graphical login screen, your graphical environment will start on the first virtual console that is not currently being used by a login session. Normally, your graphical session will replace the login prompt on the second virtual console (**tty2**). However, if that console is in use by an active text login session (not just a login prompt), the next free virtual console is used instead.

The graphical login screen continues to run on the first virtual console (**tty1**). If you are already logged in to a graphical session, and log in as another user on the graphical login screen or use the **Switch User** menu item to switch users in the graphical environment without logging out, another graphical environment will be started for that user on the next free virtual console.

When you log out of a graphical environment, it will exit and the physical console will automatically switch back to the graphical login screen on the first virtual console.

**Note**

In Red Hat Enterprise Linux 6 and 7, the graphical login screen runs on the first virtual console, but when you log in your initial graphical environment *replaces* the login screen on the first virtual console instead of starting on a new virtual console.

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

A *headless server* does not have a keyboard and display permanently connected to it. A data center may be filled with many racks of headless servers, and not providing each with a keyboard and display saves space and expense. To allow administrators to log in, a headless server might have a login prompt provided by its *serial console*, running on a serial port which is connected to a networked console server for remote access to the serial console.

The serial console would normally be used to fix the server if its own network card became misconfigured and logging in over its own network connection became impossible. Most of the time, however, headless servers are accessed by other means over the network.

Logging in over the Network

Linux users and administrators often need to get shell access to a remote system by connecting to it over the network. In a modern computing environment, many headless servers are actually virtual machines or are running as public or private cloud instances. These systems are not physical and do not have real hardware consoles. They might not even provide access to their (simulated) physical console or serial console.

In Linux, the most common way to get a shell prompt on a remote system is to use Secure Shell (SSH). Most Linux systems (including Red Hat Enterprise Linux) and macOS provide the OpenSSH command-line program **ssh** for this purpose.

In this example, a user with a shell prompt on the machine **host** uses **ssh** to log in to the remote Linux system **remotehost** as the user **remoteuser**:

```
[user@host ~]$ ssh remoteuser@remotehost
remoteuser@remotehost's password: password
[remoteuser@remotehost ~]$
```

The **ssh** command encrypts the connection to secure the communication against eavesdropping or hijacking of the passwords and content.

Some systems (such as new cloud instances) do not allow users to use a password to log in with **ssh** for tighter security. An alternative way to authenticate to a remote machine without entering a password is through *public key authentication*.

With this authentication method, users have a special identity file containing a *private key*, which is equivalent to a password, and which they keep secret. Their account on the server is configured with a matching *public key*, which does not have to be secret. When logging in, users can configure **ssh** to provide the private key and if their matching public key is installed in that account on that remote server, it will log them in without asking for a password.

In the next example, a user with a shell prompt on the machine **host** logs in to **remotehost** as **remoteuser** using **ssh**, using public key authentication. The **-i** option is used to specify the user's private key file, which is **mylab.pem**. The matching public key is already set up as an authorized key in the **remoteuser** account.

```
[user@host ~]$ ssh -i mylab.pem remoteuser@remotehost  
[remoteuser@remotehost ~]$
```

For this to work, the private key file must be readable only by the user that owns the file. In the preceding example, where the private key is in the **mylab.pem** file, the command **chmod 600 mylab.pem** could be used to ensure this. How to set file permissions is discussed in more detail in a later chapter.

Users might also have private keys configured that are tried automatically, but that discussion is beyond the scope of this section. The References at the end of this section contain links to more information on this topic.



Note

The first time you log in to a new machine, you will be prompted with a warning from **ssh** that it cannot establish the authenticity of the host:

```
[user@host ~]$ ssh -i mylab.pem remoteuser@remotehost  
The authenticity of host 'remotehost (192.0.2.42)' 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  
[remoteuser@remotehost ~]$
```

Each time you connect to a remote host with **ssh**, the remote host sends **ssh** its *host key* to authenticate itself and to help set up encrypted communication. The **ssh** command compares that against a list of saved host keys to make sure it has not changed. If the host key has changed, this might indicate that someone is trying to pretend to be that host to hijack the connection which is also known as man-in-the-middle attack. In SSH, host keys protect against man-in-the-middle attacks, these host keys are unique for each server, and they need to be changed periodically and whenever a compromise is suspected.

You will get this warning if your local machine does not have a host key saved for the remote host. If you enter **yes**, the host key that the remote host sent will be accepted and saved for future reference. Login will continue, and you should not see this message again when connecting to this host. If you enter **no**, the host key will be rejected and the connection closed.

If the local machine does have a host key saved and it does not match the one actually sent by the remote host, the connection will automatically be closed with a warning.

Logging Out

When you are finished using the shell and want to quit, you can choose one of several ways to end the session. You can enter the **exit** command to terminate the current shell session. Alternatively, finish a session by pressing **Ctrl+D**.

The following is an example of a user logging out of an SSH session:

```
[remoteuser@remotehost ~]$ exit  
logout  
Connection to remotehost closed.  
[user@host ~]$
```



References

intro(1), bash(1), console(4), pts(4), ssh(1), and ssh-keygen(1) man pages

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

For more information on OpenSSH and public key authentication, refer to the *Using secure communications between two systems with OpenSSH* chapter in the *Red Hat Enterprise Linux 8 Securing networks* guide at
https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/8/html-single/securing_networks/index#using-secure-communications-between-two-systems-with-openssh_securing-networks



Note

Instructions on how to read **man** pages and other online help documentation is included at the end of the next section.

► Quiz

Accessing the Command Line

Choose the correct answer to the following questions:

- ▶ 1. Which term describes the interpreter that executes commands typed as strings?
 - a. Command
 - b. Console
 - c. Shell
 - d. Terminal
- ▶ 2. Which term describes the visual cue that indicates an interactive shell is waiting for the user to type a command?
 - a. Argument
 - b. Command
 - c. Option
 - d. Prompt
- ▶ 3. Which term describes the name of a program to run?
 - a. Argument
 - b. Command
 - c. Option
 - d. Prompt
- ▶ 4. Which term describes the part of the command line that adjusts the behavior of a command?
 - a. Argument
 - b. Command
 - c. Option
 - d. Prompt
- ▶ 5. Which term describes the part of the command line that specifies the target that the command should operate on?
 - a. Argument
 - b. Command
 - c. Option
 - d. Prompt

► **6. Which term describes the hardware display and keyboard used to interact with a system?**

- a. Physical Console
- b. Virtual Console
- c. Shell
- d. Terminal

► **7. Which term describes one of multiple logical consoles that can each support an independent login session?**

- a. Physical Console
- b. Virtual Console
- c. Shell
- d. Terminal

► **8. Which term describes an interface that provides a display for output and a keyboard for input to a shell session?**

- a. Console
- b. Virtual Console
- c. Shell
- d. Terminal

► Solution

Accessing the Command Line

Choose the correct answer to the following questions:

- ▶ 1. Which term describes the interpreter that executes commands typed as strings?
 - a. Command
 - b. Console
 - c. Shell
 - d. Terminal
- ▶ 2. Which term describes the visual cue that indicates an interactive shell is waiting for the user to type a command?
 - a. Argument
 - b. Command
 - c. Option
 - d. Prompt
- ▶ 3. Which term describes the name of a program to run?
 - a. Argument
 - b. Command
 - c. Option
 - d. Prompt
- ▶ 4. Which term describes the part of the command line that adjusts the behavior of a command?
 - a. Argument
 - b. Command
 - c. Option
 - d. Prompt
- ▶ 5. Which term describes the part of the command line that specifies the target that the command should operate on?
 - a. Argument
 - b. Command
 - c. Option
 - d. Prompt

► **6. Which term describes the hardware display and keyboard used to interact with a system?**

- a. Physical Console
- b. Virtual Console
- c. Shell
- d. Terminal

► **7. Which term describes one of multiple logical consoles that can each support an independent login session?**

- a. Physical Console
- b. Virtual Console
- c. Shell
- d. Terminal

► **8. Which term describes an interface that provides a display for output and a keyboard for input to a shell session?**

- a. Console
- b. Virtual Console
- c. Shell
- d. Terminal

Accessing the Command Line Using the Desktop

Objectives

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

Introduction to 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 8 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 either Wayland (by default) or the legacy X Window System.

GNOME Shell provides the core user interface functions for the GNOME desktop environment. The GNOME Shell application is highly customizable. Red Hat Enterprise Linux 8 defaults GNOME Shell's look and feel to the "Standard" theme, which is used in this section. Red Hat Enterprise Linux 7 defaulted to an alternative theme named "Classic" that was closer to the look and feel of older versions of GNOME. Either theme can be selected persistently at login by clicking the gear icon next to the **Sign In** button that is available after selecting your account but before entering your password.

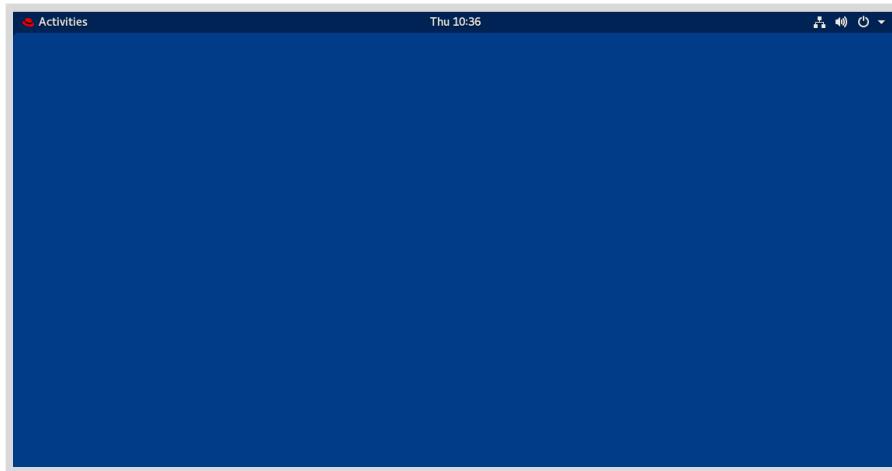
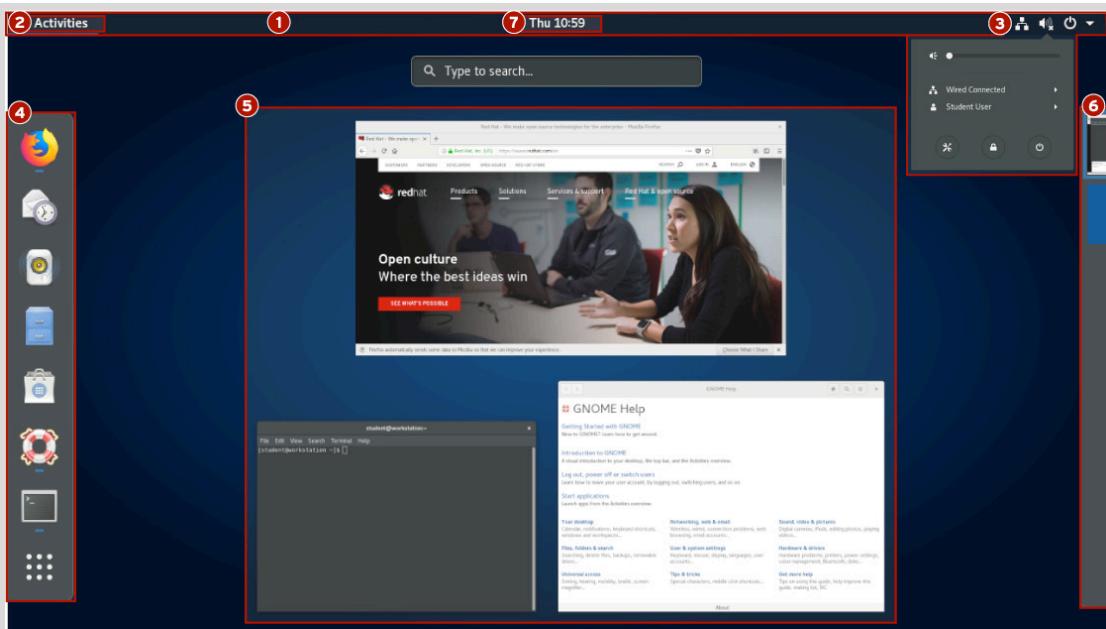


Figure 2.1: An empty GNOME 3 desktop

The first time you log in as a new user, an initial setup program runs to help configure basic account settings. Once that is complete, the GNOME Help application starts on the **Getting Started with GNOME** screen. This screen includes videos and documentation to help orient new users to the GNOME 3 environment. You can quickly start GNOME Help by clicking the **Activities** button on the left side of the top bar, and in the dash that appears on the left side of the screen, clicking the life ring buoy icon to launch it.

Parts of the GNOME Shell

The elements of the GNOME Shell include the following parts, as illustrated by this screenshot of the GNOME Shell in Activities overview mode:



- ① **Top bar:** The bar that runs along the top of the screen. It is displayed in the Activities overview and in workspaces. The top bar provides the **Activities** button, and controls for volume, networking, calendar access, and switching between keyboard input methods (if more than one is configured).
- ② **Activities overview:** This is a special mode that helps a user organize windows and start applications. The Activities overview can be entered by clicking the **Activities** button at the upper-left corner of the top bar, or by pressing the **Super** key. The **Super** key (sometimes called the **Windows** key or **Command** key), is found near the lower left corner of an IBM PC 104/105-key or Apple 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.
- ③ **System menu:** The menu in the upper-right corner on the top bar provides control to adjust the brightness of the screen, and to switch on or off the network connections. Under the submenu for the user's name are options to adjust account settings, and log out of the system. The system menu also offers buttons to open the **Settings** window, lock the screen, or shut down the system.
- ④ **Dash:** This is a configurable list of icons of the user's favorite applications, applications which are currently running, and a **grid** button at the bottom of the dash 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**.
- ⑤ **Windows overview:** An area in the center of the Activities overview which displays thumbnails of all windows active in the current workspace. This allows windows to be more easily brought to the foreground on a cluttered workspace, or to be moved to another workspace.
- ⑥ **Workspace selector:** An area to the right of the Activities overview which displays thumbnails of all active workspaces and allows workspaces to be selected and windows to be moved from one workspace to another.
- ⑦ **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 top of the screen, and a persistent indicator appears in the middle of the top bar next to the clock to inform the user of notifications have been recently received. The message tray can be opened to review these notifications by clicking

the clock on the top bar or by pressing **Super+M**. The message tray can be closed by clicking the clock on the top bar, or by pressing **Esc** or **Super+M** again.

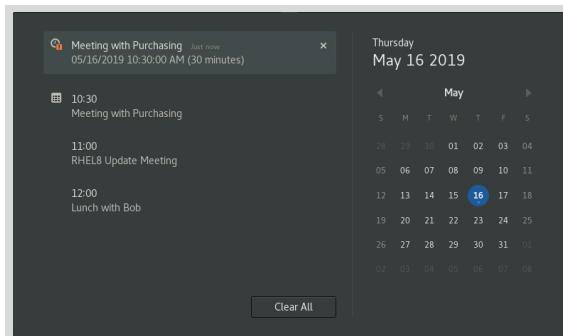


Figure 2.2: Closeup of an open message tray

You can view and edit the GNOME keyboard shortcuts used by your account. Open the system menu on the right side of the top bar. Click the **Settings** button on the bottom of the menu on the left. In the application window that opens, select **Devices → Keyboard** from the left pane. The right pane will display your current shortcut settings.



Note

Some keyboard shortcuts, such as function keys or the **Super** key, might be difficult to send to a virtual machine. This is because special keystrokes used by those shortcuts might be captured by your local operating system, or by the application that you are using to access the graphical desktop of your virtual machine.



Important

In Red Hat's current virtual training and self-paced training environments, using the **Super** key can be a little tricky. You probably cannot just use your keyboard's **Super** key because it is often not passed to the virtual machine in the classroom environment by your web browser.

At the top of your browser window that displays the interface for your virtual machine, there should be a keyboard icon on the right side. If you click that, it will open an on-screen keyboard. Clicking it again will close the on-screen keyboard.

The on-screen keyboard treats **Super** as a modifier key that is often held down while pressing another key. If you click it once, it will turn yellow indicating that the key is being held down. So, to enter **Super+M** in the on-screen keyboard, click **Super**, then click **M**.

If you just want to press and release **Super** in the on-screen keyboard, you have to click it twice. The first click "holds down" the **Super** key, and the second click releases it.

The other keys treated as modifier keys (like **Super**) by the on-screen keyboard are **Shift**, **Ctrl**, **Alt**, and **Caps**. The **Esc** and **Menu** keys are treated like normal keys and *not* modifier keys.

Workspaces

Workspaces are separate desktop screens that 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 two simple methods for switching between workspaces. One method, perhaps the fastest, is to press **Ctrl+Alt+UpArrow** or **Ctrl+Alt+DownArrow** to switch between workspaces sequentially. The second 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 workspaces using the **workspace selector** on the right side of the screen and the **windows overview** in the center of the screen.



Important

Like **Super**, in Red Hat's current virtual training and self-paced training environments, **Ctrl+Alt** key combinations are not usually passed to the virtual machine in the classroom environment by your web browser.

You can enter these key combinations to switch workspaces using the on-screen keyboard. At least two workspaces need to be in use. Open the on-screen keyboard and click **Ctrl**, **Alt**, and then either **UpArrow** or **DownArrow**.

However, in those training environments, it is generally simpler to avoid the keyboard shortcuts and the on-screen keyboard. Switch workspaces by clicking the **Activities** button and then, in the workspace selector to the right of the Activities overview, clicking on the workspace to which you want to switch.

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. The two most commonly used methods are listed below:

- 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**).
- Press the **Alt+F2** key combination to open the **Enter a Command** and enter **gnome-terminal**.

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 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 system menu on the far right of the top bar.

To lock the screen, from the system menu in the upper-right corner, click the lock button at the bottom of the menu or press **Super+L** (which might be easier to remember as **Windows+L**). The screen also locks if the graphical session is idle for a few minutes.

A **lock screen curtain** appears 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 that user's password on the **lock screen**.

To log out and end the current graphical login session, select the system menu in the upper-right corner on the top bar and select **(User) → Log Out**. A window displays that offers the option to **Cancel** or confirm the **Log Out** action.

Powering off or Rebooting the System

To shut down the system, from the system menu in the upper-right corner, click the power button at the bottom of the menu or press **Ctrl+Alt+Del**. In the dialog box that displays, you can choose to **Power Off** or **Restart** the machine, or **Cancel** the operation. If you do not make a choice, the system automatically shuts down after 60 seconds.



References

GNOME Help

- **yelp**
- GNOME Help: *Getting Started with GNOME*
- **yelp help:gnome-help/getting-started**

► Guided Exercise

Accessing the Command Line Using the Desktop

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

Outcomes

You should be able to log in to a Linux system using the GNOME 3 desktop environment, and run commands from a shell prompt in a terminal program.

Before You Begin

Ensure that the **workstation** virtual machine is running. Perform the following tasks on **workstation**.

- ▶ 1. Log in to **workstation** as **student** using **student** as the password.
 - 1.1. On **workstation**, at the GNOME login screen, click the **student** user account. Enter **student** when prompted for the password.
 - 1.2. Click **Sign In**.
- ▶ 2. Change the password for **student** from **student** to **55TurnK3y**.



Important

The **finish** script resets the password for the **student** user to **student**. The script must be executed at the end of the exercise.

- 2.1. The simplest approach is to open a Terminal window and use the **passwd** command at the shell prompt.
In the virtual learning environment with visual keyboard, press the **Super** key twice to enter **Activities** overview. Type **terminal** and then press **Enter** to start Terminal.
- 2.2. In the terminal window that displays, type **passwd** at the shell prompt. Change the student password from **student** to **55TurnK3y**.

```
[student@workstation ~]$ passwd
Changing password for user student.
Current password: student
New password: 55TurnK3y
Retype new password: 55TurnK3y
passwd: all authentication tokens updated successfully.
```

- ▶ 3. Log out and log back in as **student** using **55TurnK3y** as the password to verify the changed password.

- 3.1. Click the system menu in the upper-right corner.
 - 3.2. Select **Student User** → **Log Out**.
 - 3.3. Click **Log Out** in the confirmation dialog box that displays.
 - 3.4. At the GNOME login screen, click the **student** user account. Enter **55TurnK3y** when prompted for the password.
 - 3.5. Click **Sign In**.
- ▶ **4.** Lock the screen.
- 4.1. From the system menu in the upper-right corner, press the lock screen button at the bottom of the menu.
- ▶ **5.** Unlock the screen.
- 5.1. Press **Enter** to lift the lock screen curtain.
 - 5.2. In the **Password** field, enter **55TurnK3y** as the password.
 - 5.3. Click **Unlock**.
- ▶ **6.** Determine how to shut down **workstation** from the graphical interface, but **Cancel** the operation without shutting down the system.
- 6.1. From the system menu in the upper-right corner, click the power button at the bottom of the menu. A dialog box displays with the options to either **Restart** or **Power Off** the machine.
 - 6.2. Click **Cancel** in the dialog box that displays.

Finish

On **workstation**, run the **lab cli-desktop finish** script to complete this exercise.

```
[student@workstation ~]$ lab cli-desktop finish
```

This concludes the guided exercise.

Executing Commands Using the Bash Shell

Objectives

After completing this section, you 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 (which usually 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.

When you are ready to execute a command, press the **Enter** key. Type each command on a separate line. The command output is displayed before the next shell prompt appears.

```
[user@host]$ whoami  
user  
[user@host]$
```

If you want to type more than one command on a single line, use the semicolon (;) as a command separator. A semicolon is a member of a class of characters called *metacharacters* that have special meanings for **bash**. In this case the output of both commands will be displayed before the next shell prompt appears.

The following example shows how to combine two commands (**command1** and **command2**) on the command line.

```
[user@host]$ command1;command2
```

Examples of Simple Commands

The **date** command displays 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.

```
[user@host ~]$ date  
Sat Jan 26 08:13:50 IST 2019  
[user@host ~]$ date +%R  
08:13  
[user@host ~]$ date +%%x  
01/26/2019
```

The **passwd** command changes a user's own password. The original password for the account must be specified before a change is allowed. By default, **passwd** is configured to require a strong password, consisting of lowercase letters, uppercase letters, numbers, and symbols, and is not

Chapter 2 | Accessing the Command Line

based on a dictionary word. The superuser can use the **passwd** command to change other users' passwords.

```
[user@host ~]$ passwd
Changing password for user user.
Current 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.

```
[user@host ~]$ file /etc/passwd
/etc/passwd: ASCII text
[user@host ~]$ file /bin/passwd
/bin/passwd: setuid ELF 64-bit LSB shared object, x86-64, version 1
(SYSV), dynamically linked, interpreter /lib64/ld-linux-x86-64.so.2,
for GNU/Linux 3.2.0, BuildID[sha1]=a3637110e27e9a48dced9f38b4ae43388d32d0e4,
stripped
[user@host ~]$ file /home
/home: directory
```

Viewing the Contents of Files

One of the most simple and frequently used commands in Linux is **cat**. The **cat** command allows you to create single or multiple files, view the contents of files, concatenate the contents from multiple files, and redirect contents of the file to a terminal or files.

The example shows how to view the contents of the **/etc/passwd** file.

```
[user@host ~]$ cat /etc/passwd
root:x:0:0:root:/root:/bin/bash
bin:x:1:1:bin:/bin:/sbin/nologin
daemon:x:2:2:daemon:/sbin:/sbin/nologin
adm:x:3:4:adm:/var/adm:/sbin/nologin
...output omitted...
```

Use the following command to display the contents of multiple files.

```
[user@host ~]$ cat file1 file2
Hello World!!
Introduction to Linux commands.
```

Some files are very long and can take up more room to display than that provided by the terminal. The **cat** command does not display the contents of a file as pages. The **less** command displays one page of a file at a time and lets you scroll at your leisure.

The **less** command allows you to page forward and backward through files that are longer than can fit on one terminal window. Use the **UpArrow** key and the **DownArrow** key to scroll up and down. Press **q** to exit the command.

The **head** and **tail** commands display the beginning and end of a file, respectively. By default these commands display 10 lines of the file, 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.

```
[user@host ~]$ 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
[user@host ~]$ tail -n 3 /etc/passwd
gdm:x:42:42::/var/lib/gdm:/sbin/nologin
gnome-initial-setup:x:977:977::/run/gnome-initial-setup/:/sbin/nologin
avahi:x:70:70:Avahi mDNS/DNS-SD Stack:/var/run/avahi-daemon:/sbin/nologin
```

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

```
[user@host ~]$ wc /etc/passwd
45 102 2480 /etc/passwd
[user@host ~]$ wc -l /etc/passwd ; wc -l /etc/group
45 /etc/passwd
70 /etc/group
[user@host ~]$ wc -c /etc/group /etc/hosts
966 /etc/group
516 /etc/hosts
1482 total
```

Tab Completion

Tab completion allows a user to quickly complete commands or file names after 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.

```
[user@host ~]$ pas①Tab+Tab
passwd      paste      pasuspender
[user@host ~]$ pass②Tab
[user@host ~]$ passwd
Changing password for user user.
Current password:
```

- ① Press **Tab** twice.
- ② Press **Tab** once.

Tab completion can be used to complete file names when typing them as arguments to commands. When **Tab** is pressed, it completes as much of the file name as possible. 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 complete the command.

```
[user@host ~]$ ls /etc/pas①Tab
[user@host ~]$ ls /etc/passwd②Tab
passwd    passwd-
```

- ① ②** Press **Tab** once.

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@host ~]# useradd --①Tab+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
```

- ①** Press **Tab** twice.

Continuing a Long Command on Another Line

Commands with many options and arguments can quickly become long and are automatically wrapped by the command window when the cursor reaches the right margin. Instead, to make command readability easier, you can type a long command using more than one line.

To do this, you will use a backslash character (\), referred to as the *escape character*, to ignore the meaning of the character immediately following the backslash. You have learned that entering a newline character, by pressing the **Enter** key, tells the shell that command entry is complete and to execute the command. By escaping the newline character, the shell is told to move to a new command line without performing command execution. The shell acknowledges the request by displaying a continuation prompt, referred to as the *secondary prompt*, using the greater-than character (>) by default, on an empty new line. Commands may be continued over many lines.

```
[user@host]$ head -n 3 \
> /usr/share/dict/words \
> /usr/share/dict/linux.words
==> /usr/share/dict/words <==
1080
10-point
10th

==> /usr/share/dict/linux.words <==
1080
10-point
10th
[user@host ~]$
```



Important

The previous screen example displays how a continued command appears to a typical user. However, portraying this realism in learning materials, such as this coursebook, commonly causes confusion. New learners might mistakenly insert the extra greater-than character as part of the typed command. The shell interprets a typed greater-than character as *process redirection*, which the user did not intend. Process redirection is discussed in an upcoming chapter.

To avoid this confusion, this coursebook will not show secondary prompts in screen outputs. A user still sees the secondary prompt in their shell window, but the course material intentionally displays only characters to be typed, as demonstrated in the example below. Compare with the previous screen example.

```
[user@host]$ head -n 3 \
/usr/share/dict/words \
/usr/share/dict/linux.words
==> /usr/share/dict/words <==
1080
10-point
10th

==> /usr/share/dict/linux.words <==
1080
10-point
10th
[user@host ~]$
```

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. The **!number** command expands to the command matching the number specified. The **!string** command expands to the most recent command that begins with the string specified.

```
[user@host ~]$ history
...output omitted...
23 clear
24 who
25 pwd
26 ls /etc
27 uptime
28 ls -l
29 date
30 history
[user@host ~]$ !ls
ls -l
total 0
drwxr-xr-x. 2 user user 6 Mar 29 21:16 Desktop
...output omitted...
```

```
[user@host ~]$ !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 commands in the shell's history.

UpArrow edits the previous command in the history list. **DownArrow** edits the next command in the history list. **LeftArrow** and **RightArrow** move the cursor left and right in the current command from the history list, so that you can edit it before running it.

You can use either the **Esc+. or Alt+.** key combination to insert the last word of the previous command at the cursor's current location. Repeated use of the key combination will replace that text with the last word of even earlier commands in the history. The **Alt+.** key combination is particularly convenient because you can hold down **Alt** and press **.** repeatedly to easily go through earlier and 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+LeftArrow	Jump to the beginning of the previous word on the command line.
Ctrl+RightArrow	Jump to the end 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 new users. The other commands can be found in the **bash(1)** man page.



References

bash(1), date(1), file(1), cat(1), more(1), less(1), head(1), passwd(1), tail(1), and **wc(1)** man pages

► Quiz

Executing Commands Using the Bash Shell

Choose the correct answers to the following questions:

- ▶ 1. Which Bash shortcut or command jumps to the beginning of the previous word on the command line?
 - a. Pressing **Ctrl+LeftArrow**
 - b. Pressing **Ctrl+K**
 - c. Pressing **Ctrl+A**
 - d. **!string**
 - e. **!number**
- ▶ 2. Which Bash shortcut or command separates commands on the same line?
 - a. Pressing **Tab**
 - b. **history**
 - c. ;
 - d. **!string**
 - e. Pressing **Esc+.**
- ▶ 3. Which Bash shortcut or command is used to clear characters from the cursor to the end of the command line?
 - a. Pressing **Ctrl+LeftArrow**
 - b. Pressing **Ctrl+K**
 - c. Pressing **Ctrl+A**
 - d. ;
 - e. Pressing **Esc+.**
- ▶ 4. Which Bash shortcut or command is used to re-execute a recent command by matching the command name?
 - a. Pressing **Tab**
 - b. **!number**
 - c. **!string**
 - d. **history**
 - e. Pressing **Esc+.**

- 5. Which Bash shortcut or command is used to complete commands, file names, and options?
- a. ;
 - b. *!number*
 - c. **history**
 - d. Pressing **Tab**
 - e. Pressing **Esc+**.
- 6. Which Bash shortcut or command re-executes a specific command in the history list?
- a. Pressing **Tab**
 - b. *!number*
 - c. *!string*
 - d. **history**
 - e. Pressing **Esc+**.
- 7. Which Bash shortcut or command jumps to the beginning of the command line?
- a. *!number*
 - b. *!string*
 - c. Pressing **Ctrl+LeftArrow**
 - d. Pressing **Ctrl+K**
 - e. Pressing **Ctrl+A**
- 8. Which Bash shortcut or command displays the list of previous commands?
- a. Pressing **Tab**
 - b. *!string*
 - c. *!number*
 - d. **history**
 - e. Pressing **Esc+**.
- 9. Which Bash shortcut or command copies the last argument of previous commands?
- a. Pressing **Ctrl+K**
 - b. Pressing **Ctrl+A**
 - c. *!number*
 - d. Pressing **Esc+**.

► Solution

Executing Commands Using the Bash Shell

Choose the correct answers to the following questions:

- ▶ 1. Which Bash shortcut or command jumps to the beginning of the previous word on the command line?
 - a. Pressing **Ctrl+LeftArrow**
 - b. Pressing **Ctrl+K**
 - c. Pressing **Ctrl+A**
 - d. **!string**
 - e. **!number**
- ▶ 2. Which Bash shortcut or command separates commands on the same line?
 - a. Pressing **Tab**
 - b. **history**
 - c. ;
 - d. **!string**
 - e. Pressing **Esc+.**
- ▶ 3. Which Bash shortcut or command is used to clear characters from the cursor to the end of the command line?
 - a. Pressing **Ctrl+LeftArrow**
 - b. Pressing **Ctrl+K**
 - c. Pressing **Ctrl+A**
 - d. ;
 - e. Pressing **Esc+.**
- ▶ 4. Which Bash shortcut or command is used to re-execute a recent command by matching the command name?
 - a. Pressing **Tab**
 - b. **!number**
 - c. **!string**
 - d. **history**
 - e. Pressing **Esc+.**

- 5. Which Bash shortcut or command is used to complete commands, file names, and options?
- a. ;
 - b. !*number*
 - c. history
 - d. Pressing Tab
 - e. Pressing Esc+.
- 6. Which Bash shortcut or command re-executes a specific command in the history list?
- a. Pressing Tab
 - b. !*number*
 - c. !*string*
 - d. history
 - e. Pressing Esc+.
- 7. Which Bash shortcut or command jumps to the beginning of the command line?
- a. !*number*
 - b. !*string*
 - c. Pressing Ctrl+LeftArrow
 - d. Pressing Ctrl+K
 - e. Pressing Ctrl+A
- 8. Which Bash shortcut or command displays the list of previous commands?
- a. Pressing Tab
 - b. !*string*
 - c. !*number*
 - d. history
 - e. Pressing Esc+.
- 9. Which Bash shortcut or command copies the last argument of previous commands?
- a. Pressing Ctrl+K
 - b. Pressing Ctrl+A
 - c. !*number*
 - d. Pressing Esc+.

► Lab

Accessing the Command Line

Performance Checklist

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

Outcomes

- Successfully run simple programs using the Bash shell command line.
- Execute commands used to identify file types and display parts of text files.
- Practice using some Bash command history "shortcuts" to more efficiently repeat commands or parts of commands.

Before You Begin

Log in to **workstation** as **student** using **student** as the password.

On **workstation**, run the **lab cli-review start** script to set up a clean lab environment. The script also copies the **zcat** file to **student**'s home directory.

```
[student@workstation ~]$ lab cli-review start
```

- Use the **date** command to display the current time and date.
- Display the current time in 12-hour clock time (for example, 11:42:11 AM). Hint: The format string that displays that output is **%r**.
- What kind of file is **/home/student/zcat**? Is it readable by humans?
- Use the **wc** command and Bash shortcuts to display the size of **zcat**.
- Display the first 10 lines of **zcat**.
- Display the last 10 lines of the **zcat** file.
- Repeat the previous command exactly with three or fewer keystrokes.
- 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 number of keystrokes.
- Use the shell history to run the **date +%r** command again.

Evaluation

On workstation, run the **lab cli-review grade** script to confirm success on this exercise.

```
[student@workstation ~]$ lab cli-review grade
```

Finish

On workstation, run the **lab cli-review finish** script to complete the lab.

```
[student@workstation ~]$ lab cli-review finish
```

This concludes the lab.

► Solution

Accessing the Command Line

Performance Checklist

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

Outcomes

- Successfully run simple programs using the Bash shell command line.
- Execute commands used to identify file types and display parts of text files.
- Practice using some Bash command history "shortcuts" to more efficiently repeat commands or parts of commands.

Before You Begin

Log in to **workstation** as **student** using **student** as the password.

On **workstation**, run the **lab cli-review start** script to set up a clean lab environment. The script also copies the **zcat** file to **student**'s home directory.

```
[student@workstation ~]$ lab cli-review start
```

- Use the **date** command to display the current time and date.

```
[student@workstation ~]$ date
Thu Jan 22 10:13:04 PDT 2019
```

- Display the current time in 12-hour clock time (for example, 11:42:11 AM). Hint: The format string that displays that output is **%r**.

Use the **+%r** argument with the **date** command to display the current time in 12-hour clock time.

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

- What kind of file is **/home/student/zcat**? Is it readable by humans?

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

```
[student@workstation ~]$ file zcat
zcat: POSIX shell script, ASCII text executable
```

- Use the **wc** command and Bash shortcuts to display the size of **zcat**.

The **wc** command can be used to display the number of lines, words, and bytes in the **zcat** script. Instead of retyping the file name, use the Bash history shortcut **Esc+.** (the keys **Esc** and **.** pressed at the same time) to reuse the argument from the previous command.

```
[student@workstation ~]$ wc Esc+.  
[student@workstation ~]$ wc zcat  
51 299 1983 zcat
```

5. Display the first 10 lines of **zcat**.

The **head** command displays the beginning of the file. Try using the **Esc+.** shortcut again.

```
[student@workstation ~]$ head Esc+.  
[student@workstation ~]$ head zcat  
#!/bin/sh  
# Uncompress files to standard output.  
  
# Copyright (C) 2007, 2010-2018 Free Software Foundation, Inc.  
  
# This program is free software; you can redistribute it and/or modify  
# it under the terms of the GNU General Public License as published by  
# the Free Software Foundation; either version 3 of the License, or  
# (at your option) any later version.
```

6. Display the last 10 lines of the **zcat** file.

Use the **tail** command to display the last 10 lines of the **zcat** file.

```
[student@workstation ~]$ tail Esc+.  
[student@workstation ~]$ tail zcat  
With no FILE, or when FILE is -, read standard input.  
  
Report bugs to <bug-gzip@gnu.org>."  
  
case $1 in  
--help) printf '%s\n' "$usage" || exit 1;;  
--version) printf '%s\n' "$version" || exit 1;;  
esac  
  
exec gzip -cd "$@"
```

7. Repeat the previous command exactly with three or fewer keystrokes.

Repeat the previous command exactly. Either press the **UpArrow** key once to scroll back through the command history one command and then press **Enter** (uses two keystrokes),

or enter the shortcut command **!!** and then press **Enter** (uses three keystrokes) to run the most recent command in the command history . (Try both.)

```
[student@workstation]$ !!
tail zcat
With no FILE, or when FILE is -, read standard input.

Report bugs to <bug-gzip@gnu.org>.

case $1 in
--help)    printf '%s\n' "$usage"  || exit 1;;
--version) printf '%s\n' "$version" || exit 1;;
esac

exec gzip -cd "$@"
```

8. 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 number of keystrokes.

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

```
[student@workstation ~]$ tail -n 20 zcat
-l, --list      list compressed file contents
-q, --quiet     suppress all warnings
-r, --recursive operate recursively on directories
-S, --suffix=SUF use suffix SUF on compressed files
--synchronous   synchronous output (safer if system crashes, but slower)
-t, --test       test compressed file integrity
-v, --verbose    verbose mode
--help          display this help and exit
--version       display version information and exit
```

With no FILE, or when FILE is -, read standard input.

```
Report bugs to <bug-gzip@gnu.org>.

case $1 in
--help)    printf '%s\n' "$usage"  || exit 1; exit;;
--version) printf '%s\n' "$version" || exit 1; exit;;
esac

exec gzip -cd "$@"
```

9. Use the shell history to run the **date +%r** command again.

Use the **history** command to display the list of previous commands and to identify the specific **date** command to be executed. Use **!number** to run the command, where *number* is the command number to use from the output of the **history** command.

Note that your shell history may be different from the following example. Determine the command number to use based on the output of your own **history** command.

```
[student@workstation ~]$ history
1 date
2 date +%r
3 file zcat
4 wc zcat
5 head zcat
6 tail zcat
7 tail -n 20 zcat
8 history
[student@workstation ~]$ !2
date +%r
10:49:56 AM
```

Evaluation

On workstation, run the **lab cli-review grade** script to confirm success on this exercise.

```
[student@workstation ~]$ lab cli-review grade
```

Finish

On workstation, run the **lab cli-review finish** script to complete the lab.

```
[student@workstation ~]$ lab cli-review finish
```

This concludes the lab.

Summary

In this chapter, you learned:

- The Bash shell is a command interpreter that prompts interactive users to specify Linux commands.
- Many commands have a **--help** option that displays a usage message or screen.
- Using workspaces makes it easier to organize multiple application windows.
- The **Activities** button located at the upper-left corner of the top bar provides an overview mode that helps a user organize windows and start applications.
- The **file** command scans the beginning of a file's contents and displays what type it is.
- The **head** and **tail** commands display the beginning and end of a file, respectively.
- You can use **Tab** completion to complete file names when typing them as arguments to commands.

Chapter 3

Managing Files From the Command Line

Goal

Copy, move, create, delete, and organize files while working from the Bash shell.

Objectives

- Describe how Linux organizes files, and the purposes of various directories in the file-system hierarchy.
- Specify the location of files relative to the current working directory and by absolute location, determine and change your working directory, and list the contents of directories.
- Create, copy, move, and remove files and directories.
- Make multiple file names reference the same file using hard links and symbolic (or "soft") links.
- Efficiently run commands affecting many files by using pattern matching features of the Bash shell.

Sections

- Describing Linux File-system Hierarchy Concepts (and Quiz)
- Specifying Files by Name (and Quiz)
- Managing Files Using Command-line Tools (and Guided Exercise)
- Making Links Between Files (and Guided Exercise)
- Matching File Names Using Shell Expansions (and Quiz)

Lab

Managing Files from the Command Line

Describing Linux File System Hierarchy Concepts

Objectives

After completing this section, you should be able to describe how Linux organizes files, and the purposes of various directories in the file-system hierarchy.

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.

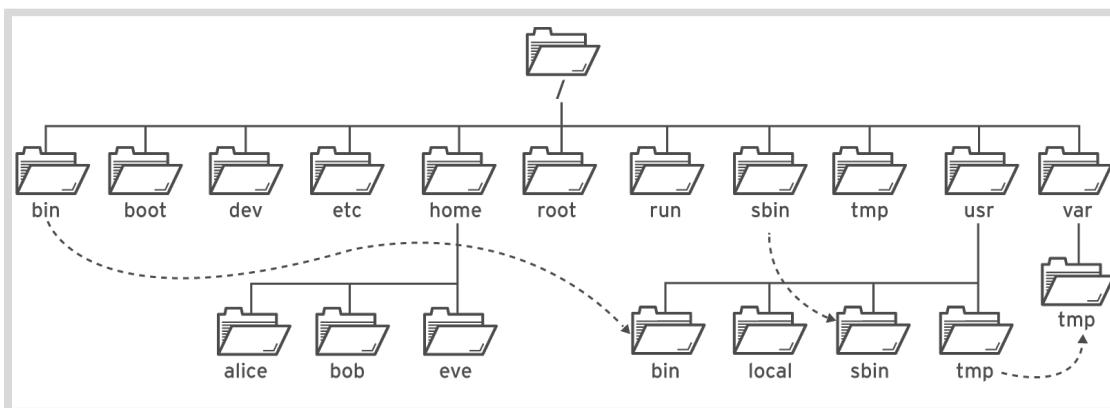


Figure 3.1: Significant file-system directories in Red Hat Enterprise Linux 8

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, you could refer to that directory as `/etc`. Likewise, if the `/etc` directory contained a file named `issue`, you 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 help to describe file-system directory contents:

- *static* content remains unchanged until explicitly edited or reconfigured.
- *dynamic* or *variable* content may be modified or appended by active processes.
- *persistent* content remains after a reboot, like configuration settings.
- *runtime* content is process- or system-specific content that is deleted by a 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 read-only program data. Important subdirectories include: <ul style="list-style-type: none"> • /usr/bin: User commands. • /usr/sbin: System administration commands. • /usr/local: Locally customized software.
/etc	Configuration files specific to this system.
/var	Variable data specific to this system that should persist between boots. Files that dynamically change, such as databases, cache directories, log files, printer-spoiled 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 earlier versions of Red Hat Enterprise Linux.
/home	<i>Home directories</i> are 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 have not been accessed, changed, or modified for 10 days 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> that are used by the system to access hardware.



Important

In Red Hat Enterprise Linux 7 and later, four older directories in `/` have identical contents to their counterparts located in `/usr`:

- `/bin` and `/usr/bin`
- `/sbin` and `/usr/sbin`
- `/lib` and `/usr/lib`
- `/lib64` and `/usr/lib64`

In earlier versions of Red Hat Enterprise Linux, these were distinct directories containing different sets of files.

In Red Hat Enterprise Linux 7 and later, the directories in `/` are symbolic links to the matching directories in `/usr`.



References

`hier(7)` man page

The UsrMove feature page from Fedora 17

<https://fedoraproject.org/wiki/Features/UsrMove>

► Quiz

Describing Linux File System Hierarchy Concepts

Choose the correct answers to the following questions:

► 1. Which directory contains persistent, system-specific configuration data?

- a. /etc
- b. /root
- c. /run
- d. /usr

► 2. Which directory is the top of the system's file system hierarchy?

- a. /etc
- b. /
- c. /home/root
- d. /root

► 3. Which directory contains user home directories?

- a. /
- b. /home
- c. /root
- d. /user

► 4. Which directory contains temporary files?

- a. /tmp
- b. /trash
- c. /run
- d. /var

► 5. Which directory contains dynamic data, such as for databases and websites?

- a. /etc
- b. /run
- c. /usr
- d. /var

► 6. Which directory is the administrative superuser's home directory?

- a. /etc
- b. /
- c. /home/root
- d. /root

► **7. Which directory contains regular commands and utilities?**

- a. /commands
- b. /run
- c. /usr/bin
- d. /usr/sbin

► **8. Which directory contains non-persistent process runtime data?**

- a. /tmp
- b. /etc
- c. /run
- d. /var

► **9. Which directory contains installed software programs and libraries?**

- a. /etc
- b. /lib
- c. /usr
- d. /var

► Solution

Describing Linux File System Hierarchy Concepts

Choose the correct answers to the following questions:

► 1. Which directory contains persistent, system-specific configuration data?

- a. /etc
- b. /root
- c. /run
- d. /usr

► 2. Which directory is the top of the system's file system hierarchy?

- a. /etc
- b. /
- c. /home/root
- d. /root

► 3. Which directory contains user home directories?

- a. /
- b. /home
- c. /root
- d. /user

► 4. Which directory contains temporary files?

- a. /tmp
- b. /trash
- c. /run
- d. /var

► 5. Which directory contains dynamic data, such as for databases and websites?

- a. /etc
- b. /run
- c. /usr
- d. /var

► 6. Which directory is the administrative superuser's home directory?

- a. /etc
- b. /
- c. /home/root
- d. /root

► **7. Which directory contains regular commands and utilities?**

- a. /commands
- b. /run
- c. /usr/bin
- d. /usr/sbin

► **8. Which directory contains non-persistent process runtime data?**

- a. /tmp
- b. /etc
- c. /run
- d. /var

► **9. Which directory contains installed software programs and libraries?**

- a. /etc
- b. /lib
- c. /usr
- d. /var

Specifying Files by Name

Objectives

After completing this section, you should be able to specify the location of files relative to the current working directory and by absolute location, determine and change the working directory, and list the contents of directories.

Absolute Paths and Relative Paths

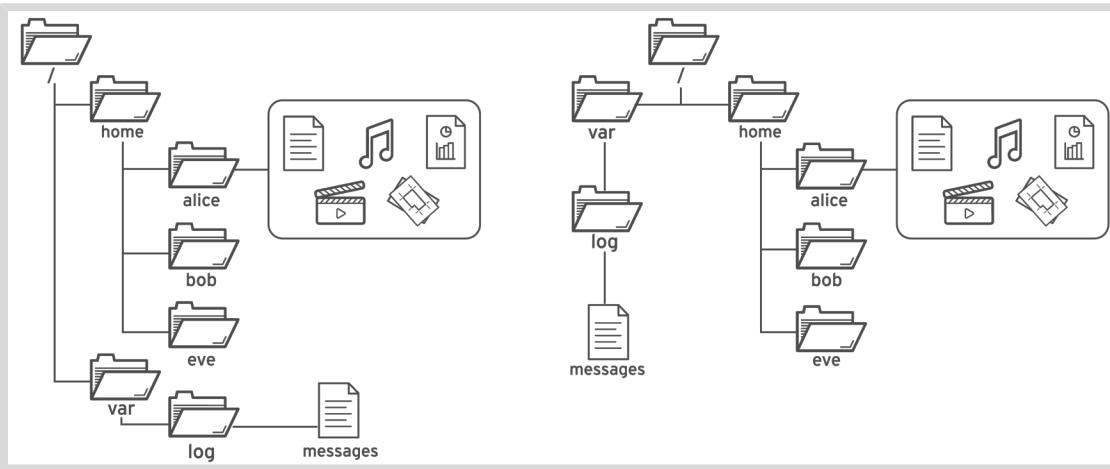


Figure 3.2: The common file browser view (left) is equivalent to the top-down view (right).

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. Directories, also called *folders*, contain other files and other subdirectories. They can be referenced in the same manner as files.



Important

A space character is acceptable as part of a Linux file name. However, spaces are also used by the shell to separate options and arguments on the command line. If you enter a command that includes a file that has a space in its name, the shell can misinterpret the command and assume that you want to start a new file name or other argument at the space.

It is possible to avoid this by putting file names in quotes. However, if you do not need to use spaces in file names, it can be simpler to simply avoid using them.

Absolute Paths

An *absolute path* is a *fully qualified* name, specifying the files exact location in the file system hierarchy. It begins at the root (/) directory and specifies each subdirectory that must be traversed to reach the specific 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 relative to the current working directory for your shell prompt.

The Current Working Directory and Relative Paths

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.

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 the first character is a relative path name. A user in the `/var` directory could refer to the message log file relatively as `log/messages`.

Linux file systems, including, but not limited to, ext4, XFS, GFS2, and GlusterFS, are case-sensitive. Creating `FileCase.txt` and `filecase.txt` in the same directory results in two unique files.

Non-Linux file systems might work differently. For example, VFAT, Microsoft's NTFS, and Apple's HFS+ have *case preserving* behavior. Although these file systems are *not* case-sensitive, they do display file names with the original capitalization used when the file was created. Therefore, if you tried to make the files in the preceding example on a VFAT file system, both names would be treated as pointing to the same file instead of two different files.

Navigating Paths

The `pwd` command displays the full path name of the current working directory for that shell. This can help you determine the syntax to reach files using relative path names. The `ls` command lists directory contents for the specified directory or, if no directory is given, for the current working directory.

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

Use the `cd` command to change your shell's current working directory. If you do not specify any arguments to the command, it will change to your home directory.

In the following example, a mixture of absolute and relative paths are used with the `cd` command to change the current working directory for the shell.

```
[user@host ~]$ pwd
/home/user
[user@host ~]$ cd Videos
[user@host Videos]$ pwd
/home/user/Videos
[user@host Videos]$ cd /home/user/Documents
[user@host Documents]$ pwd
/home/user/Documents
[user@host Documents]$ cd
```

```
[user@host ~]$ pwd  
/home/user  
[user@host ~]$
```

As you can see in the preceding example, the default shell prompt also displays the last component of the absolute path to the current working directory. For example, for **/home/user/Videos**, only **Videos** displays. The prompt displays the tilde character (~) when your current working directory is your 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, because "touching" a file name that does not exist causes the file to be created. In the following example, the **touch** command creates practice files in the **Documents** and **Videos** subdirectories.

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

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

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

```
drwxr-xr-x. 2 user user 4096 Jan  9 15:00 Public
drwxr-xr-x. 2 user user 4096 Jan  9 15:00 Templates
drwxr-xr-x. 2 user user 4096 Jan  9 15:00 Videos
[user@host ~]$
```

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. You will discover their usefulness when you start using file management commands.



Important

File names beginning with a dot (.) indicate *hidden* files; you cannot see them in the 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*.

```
[user@host ~]$ 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
[user@host ~]$
```

The **cd** command has many options. A few are so useful as to be worth practicing early and using often. The command **cd -** changes to the previous directory; where the user was previously to the current directory. The following example illustrates this behavior, alternating between two directories, which is useful when processing a series of similar tasks.

```
[user@host ~]$ cd Videos
[user@host Videos]$ pwd
/home/user/Videos
[user@host Videos]$ cd /home/user/Documents
[user@host Documents]$ pwd
/home/user/Documents
```

```
[user@host Documents]$ cd -
[user@host Videos]$ pwd
/home/user/Videos
[user@host Videos]$ cd -
[user@host Documents]$ pwd
/home/user/Documents
[user@host Documents]$ cd -
[user@host Videos]$ pwd
/home/user/Videos
[user@host Videos]$ cd
[user@host ~]$
```

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.

```
[user@host Videos]$ pwd
/home/user/Videos
[user@host Videos]$ cd .
[user@host Videos]$ pwd
/home/user/Videos
[user@host Videos]$ cd ..
[user@host ~]$ pwd
/home/user
[user@host ~]$ cd ..
[user@host home]$ pwd
/home
[user@host home]$ cd ..
[user@host /]$ pwd
/
[user@host /]$ cd
[user@host ~]$ pwd
/home/user
[user@host ~]$
```



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/>

► Quiz

Specifying Files by Name

Choose the correct answers to the following questions:

- ▶ 1. Which command is used to return to the current user's home directory, assuming the current working directory is /tmp and their home directory is /home/user?
 - a. `cd`
 - b. `cd ..`
 - c. `cd .`
 - d. `cd *`
 - e. `cd /home`

- ▶ 2. Which command displays the absolute path name of the current location?
 - a. `cd`
 - b. `pwd`
 - c. `ls ~`
 - d. `ls -d`

- ▶ 3. Which command will always return you to the working directory used prior to the current working directory?
 - a. `cd -`
 - b. `cd -p`
 - c. `cd ~`
 - d. `cd ..`

- ▶ 4. Which command will always change the working directory up two levels from the current location?
 - a. `cd ~`
 - b. `cd ../`
 - c. `cd ../../..`
 - d. `cd -u2`

- ▶ 5. Which command lists files in the current location, using a long format, and including hidden files?
 - a. `llong ~`
 - b. `ls -a`
 - c. `ls -l`
 - d. `ls -al`

- ▶ **6. Which command will always change the working directory to /bin?**
 - a. **cd bin**
 - b. **cd /bin**
 - c. **cd ~bin**
 - d. **cd -bin**
- ▶ **7. Which command will always change the working directory to the parent of the current location?**
 - a. **cd ~**
 - b. **cd ..**
 - c. **cd ../../..**
 - d. **cd -u1**
- ▶ **8. Which command will change the working directory to /tmp if the current working directory is /home/student?**
 - a. **cd tmp**
 - b. **cd ..**
 - c. **cd ../../tmp**
 - d. **cd ~tmp**

► Solution

Specifying Files by Name

Choose the correct answers to the following questions:

- ▶ 1. Which command is used to return to the current user's home directory, assuming the current working directory is /tmp and their home directory is /home/user?
 - a. **cd**
 - b. **cd ..**
 - c. **cd .**
 - d. **cd ***
 - e. **cd /home**

- ▶ 2. Which command displays the absolute path name of the current location?
 - a. **cd**
 - b. **pwd**
 - c. **ls ~**
 - d. **ls -d**

- ▶ 3. Which command will always return you to the working directory used prior to the current working directory?
 - a. **cd -**
 - b. **cd -p**
 - c. **cd ~**
 - d. **cd ..**

- ▶ 4. Which command will always change the working directory up two levels from the current location?
 - a. **cd ~**
 - b. **cd ../**
 - c. **cd ../../..**
 - d. **cd -u2**

- ▶ 5. Which command lists files in the current location, using a long format, and including hidden files?
 - a. **llong ~**
 - b. **ls -a**
 - c. **ls -l**
 - d. **ls -al**

- ▶ **6. Which command will always change the working directory to /bin?**
 - a. `cd bin`
 - b. `cd /bin`
 - c. `cd ~bin`
 - d. `cd -bin`
- ▶ **7. Which command will always change the working directory to the parent of the current location?**
 - a. `cd ~`
 - b. `cd ..`
 - c. `cd ../..`
 - d. `cd -u1`
- ▶ **8. Which command will change the working directory to /tmp if the current working directory is /home/student?**
 - a. `cd tmp`
 - b. `cd ..`
 - c. `cd ../../tmp`
 - d. `cd ~tmp`

Managing Files Using Command-line Tools

Objectives

After completing this section, you should be able to create, copy, move, and remove files and directories.

Command-line File Management

To manage files, you need to be able to create, remove, copy, and move them. You also need to organize them logically into directories, which you also need to be able to create, remove, copy, and move.

The following table summarizes some of the most common file management commands. The remainder of this section will discuss ways to use these commands in more detail.

Common file management commands

Activity	Command Syntax
Create a directory	<code>mkdir directory</code>
Copy a file	<code>cp file new-file</code>
Copy a directory and its contents	<code>cp -r directory new-directory</code>
Move or rename a file or directory	<code>mv file new-file</code>
Remove a file	<code>rm file</code>
Remove a directory containing files	<code>rm -r directory</code>
Remove an empty directory	<code>rmdir directory</code>

Creating Directories

The `mkdir` command creates one or more directories or subdirectories. It takes as arguments a list of paths to the directories you want to create.

The `mkdir` command will fail with an error if the directory already exists, or if you are trying to create a subdirectory in a directory that does not exist. The `-p (parent)` option creates missing parent directories for the requested destination. Use the `mkdir -p` command with caution, because spelling mistakes can create unintended directories without generating error messages.

In the following example, pretend that you are trying to create a directory in the `Videos` directory named `Watched`, but you accidentally left off the letter "s" in `Videos` in your `mkdir` command.

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

The `mkdir` command failed because **Videos** was misspelled and the directory **Video** does not exist. If you had used the `mkdir` command with the `-p` option, the directory **Video** would be created, which was not what you had intended, and the subdirectory **Watched** would be created in that incorrect directory.

After correctly spelling the **Videos** parent directory, creating the **Watched** subdirectory will succeed.

```
[user@host ~]$ mkdir Videos/Watched
[user@host ~]$ ls -R Videos
Videos/:
blockbuster1.ogg blockbuster2.ogg Watched

Videos/Watched:
```

In the following example, files and directories are organized beneath the `/home/user/Documents` directory. Use the `mkdir` command and a space-delimited list of the directory names to create multiple directories.

```
[user@host ~]$ cd Documents
[user@host Documents]$ mkdir ProjectX ProjectY
[user@host Documents]$ ls
ProjectX ProjectY
```

Use the `mkdir -p` command and space-delimited relative paths for each of the subdirectory names to create multiple parent directories with subdirectories.

```
[user@host Documents]$ mkdir -p Thesis/Chapter1 Thesis/Chapter2 Thesis/Chapter3
[user@host Documents]$ cd
[user@host ~]$ ls -R Videos Documents
Documents:
ProjectX ProjectY Thesis

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:
```

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

Copying Files

The **cp** command copies a file, creating a new file either in the current directory or in a specified directory. It can also copy multiple files to a directory.



Warning

If the destination file already exists, the **cp** command overwrites the file.

```
[user@host ~]$ cd Videos
[user@host Videos]$ cp blockbuster1.ogg blockbuster3.ogg
[user@host Videos]$ ls -l
total 0
-rw-rw-r--. 1 user user    0 Feb  8 16:23 blockbuster1.ogg
-rw-rw-r--. 1 user user    0 Feb  8 16:24 blockbuster2.ogg
-rw-rw-r--. 1 user user    0 Feb  8 16:34 blockbuster3.ogg
drwxrwxr-x. 2 user user 4096 Feb  8 16:05 Watched
[user@host 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. If a file with the same name exists in the target directory, the existing file is overwritten. By default, the **cp** does not copy directories; it ignores them.

In the following example, two directories are listed, **Thesis** and **ProjectX**. Only the last argument, **ProjectX** is valid as a destination. The **Thesis** directory is ignored.

```
[user@host Videos]$ cd ..../Documents
[user@host Documents]$ cp thesis_chapter1.odf thesis_chapter2.odf Thesis ProjectX
cp: omitting directory `Thesis'
[user@host Documents]$ ls Thesis ProjectX
ProjectX:
thesis_chapter1.odf  thesis_chapter2.odf

Thesis:
Chapter1  Chapter2  Chapter3
```

In the first **cp** command, the **Thesis** directory failed to copy, but the **thesis_chapter1.odf** and **thesis_chapter2.odf** files succeeded.

If you want to copy a file to the current working directory, you can use the special `.` directory:

```
[user@host ~]$ cp /etc/hostname .
[user@host ~]$ cat hostname
host.example.com
[user@host ~]$
```

Use the copy command with the **-r** (recursive) option, to copy the **Thesis** directory and its contents to the **ProjectX** directory.

```
[user@host Documents]$ cp -r Thesis ProjectX
[user@host Documents]$ ls -R ProjectX
ProjectX:
Thesis  thesis_chapter1.odf  thesis_chapter2.odf

ProjectX/Thesis:
Chapter1  Chapter2  Chapter3

ProjectX/Thesis/Chapter1:

ProjectX/Thesis/Chapter2:
thesis_chapter2.odf

ProjectX/Thesis/Chapter3:
```

Moving Files

The **mv** command moves files from one location to another. If you think of the absolute path to a file as its full name, moving a file is effectively the same as renaming a file. File contents remain unchanged.

Use the **mv** command to *rename* a file.

```
[user@host Videos]$ cd ../Documents
[user@host Documents]$ ls -l thesis*
-rw-rw-r-- 1 user user 0 Feb  6 21:16 thesis_chapter1.odf
-rw-rw-r-- 1 user user 0 Feb  6 21:16 thesis_chapter2.odf
[user@host Documents]$ mv thesis_chapter2.odf thesis_chapter2_reviewed.odf
[user@host Documents]$ ls -l thesis*
-rw-rw-r-- 1 user user 0 Feb  6 21:16 thesis_chapter1.odf
-rw-rw-r-- 1 user user 0 Feb  6 21:16 thesis_chapter2_reviewed.odf
```

Use the **mv** command to *move* a file to a different directory.

```
[user@host Documents]$ ls Thesis/Chapter1
[user@host Documents]$
[user@host Documents]$ mv thesis_chapter1.odf Thesis/Chapter1
[user@host Documents]$ ls Thesis/Chapter1
thesis_chapter1.odf
[user@host Documents]$ ls -l thesis*
-rw-rw-r-- 1 user user 0 Feb  6 21:16 thesis_chapter2_reviewed.odf
```

Removing Files and Directories

The **rm** command removes files. By default, **rm** will not remove directories that contain files, unless you add the **-r** or **--recursive** option.



Important

There is no command-line undelete feature, nor a "trash bin" from which you can restore files staged for deletion.

It is a good idea to verify your current working directory before removing a file or directory.

```
[user@host Documents]$ pwd  
/home/student/Documents
```

Use the **rm** command to remove a single file from your working directory.

```
[user@host Documents]$ ls -l thesis*  
-rw-rw-r--. 1 user user 0 Feb  6 21:16 thesis_chapter2_reviewed.odf  
[user@host Documents]$ rm thesis_chapter2_reviewed.odf  
[user@host Documents]$ ls -l thesis*  
ls: cannot access 'thesis*': No such file or directory
```

If you attempt to use the **rm** command to remove a directory without using the **-r** option, the command will fail.

```
[user@host Documents]$ rm Thesis/Chapter1  
rm: cannot remove `Thesis/Chapter1': Is a directory
```

Use the **rm -r** command to remove a subdirectory and its contents.

```
[user@host Documents]$ ls -R Thesis  
Thesis/:  
Chapter1 Chapter2 Chapter3  
  
Thesis/Chapter1:  
thesis_chapter1.odf  
  
Thesis/Chapter2:  
thesis_chapter2.odf  
  
Thesis/Chapter3:  
[user@host Documents]$ rm -r Thesis/Chapter1  
[user@host Documents]$ ls -l Thesis  
total 8  
drwxrwxr-x. 2 user user 4096 Feb 11 12:47 Chapter2  
drwxrwxr-x. 2 user user 4096 Feb 11 12:48 Chapter3
```

The **rm -r** command traverses each subdirectory first, individually removing their files before removing each directory. You can use the **rm -ri** command to interactively prompt for confirmation before deleting. This is essentially the opposite of using the **-f** option, which forces the removal without prompting the user for confirmation.

```
[user@host 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  
[user@host Documents]$
```

**Warning**

If you specify both the **-i** and **-f** options, the **-f** option takes priority and you will not be prompted for confirmation before **rm** deletes files.

In the following example, the **rmdir** command only removes the directory that is empty. Just like the earlier example, you must use the **rm -r** command to remove a directory that contains content.

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

**Note**

The **rm** command with no options cannot remove an empty directory. You must use the **rmdir** command, **rm -d** (which is equivalent to **rmdir**), or **rm -r**.

**References**

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

► Guided Exercise

Managing Files Using Command-line Tools

In this exercise you will create, organize, copy, and remove files and directories.

Outcomes

You should be able to create, organize, copy, and remove files and directories.

Before You Begin

Log in as the **student** user on **workstation** using **student** as the password.

On **workstation**, run the **lab files-manage start** command. This command runs a start script that determines if the **servera** machine is reachable on the network.

```
[student@workstation ~]$ lab files-manage start
```

- 1. Use the **ssh** command to log in to **servera** as the **student** user. The systems are configured to use SSH keys for authentication, therefore a password is not required.

```
[student@workstation ~]$ ssh student@servera  
...output omitted...  
[student@servera ~]$
```

- 2. In the **student** user's home directory, use the **mkdir** command to create three subdirectories: **Music**, **Pictures**, and **Videos**.

```
[student@servera ~]$ mkdir Music Pictures Videos
```

- 3. Continuing in the **student** user's home directory, use the **touch** command to create sets of empty practice files to use during this lab.

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

```
total 0
-rw-rw-r-- 1 student student 0 Feb  4 18:23 film1.avi
-rw-rw-r-- 1 student student 0 Feb  4 18:23 film2.avi
-rw-rw-r-- 1 student student 0 Feb  4 18:23 film3.avi
-rw-rw-r-- 1 student student 0 Feb  4 18:23 film4.avi
-rw-rw-r-- 1 student student 0 Feb  4 18:23 film5.avi
-rw-rw-r-- 1 student student 0 Feb  4 18:23 film6.avi
drwxrwxr-x. 2 student student 6 Feb  4 18:23 Music
drwxrwxr-x. 2 student student 6 Feb  4 18:23 Pictures
-rw-rw-r-- 1 student student 0 Feb  4 18:23 snap1.jpg
-rw-rw-r-- 1 student student 0 Feb  4 18:23 snap2.jpg
-rw-rw-r-- 1 student student 0 Feb  4 18:23 snap3.jpg
-rw-rw-r-- 1 student student 0 Feb  4 18:23 snap4.jpg
-rw-rw-r-- 1 student student 0 Feb  4 18:23 snap5.jpg
-rw-rw-r-- 1 student student 0 Feb  4 18:23 snap6.jpg
-rw-rw-r-- 1 student student 0 Feb  4 18:23 song1.mp3
-rw-rw-r-- 1 student student 0 Feb  4 18:23 song2.mp3
-rw-rw-r-- 1 student student 0 Feb  4 18:23 song3.mp3
-rw-rw-r-- 1 student student 0 Feb  4 18:23 song4.mp3
-rw-rw-r-- 1 student student 0 Feb  4 18:23 song5.mp3
-rw-rw-r-- 1 student student 0 Feb  4 18:23 song6.mp3
drwxrwxr-x. 2 student student 6 Feb  4 18:23 Videos
```

- ▶ 4. Continuing in the **student** user's home directory, move the song files to the **Music** subdirectory, the snapshot files to the **Pictures** subdirectory, and the movie files to the **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@servera ~]$ mv song1.mp3 song2.mp3 song3.mp3 song4.mp3 \
song5.mp3 song6.mp3 Music
[student@servera ~]$ mv snap1.jpg snap2.jpg snap3.jpg snap4.jpg \
snap5.jpg snap6.jpg Pictures
[student@servera ~]$ mv film1.avi film2.avi film3.avi film4.avi \
film5.avi film6.avi Videos
[student@servera ~]$ ls -l Music Pictures Videos
Music:
total 0
-rw-rw-r-- 1 student student 0 Feb  4 18:23 song1.mp3
-rw-rw-r-- 1 student student 0 Feb  4 18:23 song2.mp3
-rw-rw-r-- 1 student student 0 Feb  4 18:23 song3.mp3
-rw-rw-r-- 1 student student 0 Feb  4 18:23 song4.mp3
-rw-rw-r-- 1 student student 0 Feb  4 18:23 song5.mp3
-rw-rw-r-- 1 student student 0 Feb  4 18:23 song6.mp3

Pictures:
total 0
-rw-rw-r-- 1 student student 0 Feb  4 18:23 snap1.jpg
-rw-rw-r-- 1 student student 0 Feb  4 18:23 snap2.jpg
-rw-rw-r-- 1 student student 0 Feb  4 18:23 snap3.jpg
-rw-rw-r-- 1 student student 0 Feb  4 18:23 snap4.jpg
-rw-rw-r-- 1 student student 0 Feb  4 18:23 snap5.jpg
-rw-rw-r-- 1 student student 0 Feb  4 18:23 snap6.jpg
```

```
Videos:
total 0
-rw-rw-r--. 1 student student 0 Feb  4 18:23 film1.avi
-rw-rw-r--. 1 student student 0 Feb  4 18:23 film2.avi
-rw-rw-r--. 1 student student 0 Feb  4 18:23 film3.avi
-rw-rw-r--. 1 student student 0 Feb  4 18:23 film4.avi
-rw-rw-r--. 1 student student 0 Feb  4 18:23 film5.avi
-rw-rw-r--. 1 student student 0 Feb  4 18:23 film6.avi
```

- 5. Continuing in the **student** user's home directory, create three subdirectories for organizing your files into projects. Name the subdirectories **friends**, **family**, and **work**. Use a single command to create all three subdirectories at the same time.

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

```
[student@servera ~]$ mkdir friends family work
[student@servera ~]$ ls -l
total 0
drwxrwxr-x. 2 student student 6 Feb  4 18:38 family
drwxrwxr-x. 2 student student 6 Feb  4 18:38 friends
drwxrwxr-x. 2 student student 108 Feb  4 18:36 Music
drwxrwxr-x. 2 student student 108 Feb  4 18:36 Pictures
drwxrwxr-x. 2 student student 108 Feb  4 18:36 Videos
drwxrwxr-x. 2 student student 6 Feb  4 18:38 work
```

- 6. Copy a selection of new files to the project directories **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 to this directory. Keep in mind that you are making copies, therefore the original files will remain in their original locations after the files are copied to the project directories.

- Copy files (all types) containing the numbers 1 and 2 in to the **friends** subdirectory.
- Copy files (all types) containing the numbers 3 and 4 in to the **family** subdirectory.

When copying files from multiple locations into a single location, Red Hat recommends that you change to the destination directory prior to copying the files. Use the simplest path syntax, absolute or relative, to reach the source for each file management task.

```
[student@servera ~]$ cd friends
[student@servera friends]$ cp ~/Music/song1.mp3 ~/Music/song2.mp3 \
~/Pictures/snap1.jpg ~/Pictures/snap2.jpg ~/Videos/film1.avi \
~/Videos/film2.avi .
[student@servera friends]$ ls -l
total 0
-rw-rw-r--. 1 student student 0 Feb  4 18:42 film1.avi
-rw-rw-r--. 1 student student 0 Feb  4 18:42 film2.avi
-rw-rw-r--. 1 student student 0 Feb  4 18:42 snap1.jpg
-rw-rw-r--. 1 student student 0 Feb  4 18:42 snap2.jpg
-rw-rw-r--. 1 student student 0 Feb  4 18:42 song1.mp3
-rw-rw-r--. 1 student student 0 Feb  4 18:42 song2.mp3
[student@servera friends]$ cd ../family
[student@servera family]$ cp ~/Music/song3.mp3 ~/Music/song4.mp3 \
~/Pictures/snap3.jpg ~/Pictures/snap4.jpg ~/Videos/film3.avi \
```

```
~/Videos/film4.avi .
[student@servera family]$ ls -l
total 0
-rw-rw-r-- 1 student student 0 Feb  4 18:44 film3.avi
-rw-rw-r-- 1 student student 0 Feb  4 18:44 film4.avi
-rw-rw-r-- 1 student student 0 Feb  4 18:44 snap3.jpg
-rw-rw-r-- 1 student student 0 Feb  4 18:44 snap4.jpg
-rw-rw-r-- 1 student student 0 Feb  4 18:44 song3.mp3
-rw-rw-r-- 1 student student 0 Feb  4 18:44 song4.mp3
```

- 7. For your work project, create additional copies.

```
[student@servera family]$ cd ../work
[student@servera work]$ cp ~/Music/song5.mp3 ~/Music/song6.mp3 \
~/Pictures/snap5.jpg ~/Pictures/snap6.jpg \
~/Videos/film5.avi ~/Videos/film6.avi .
[student@servera work]$ ls -l
total 0
-rw-rw-r-- 1 student student 0 Feb  4 18:48 film5.avi
-rw-rw-r-- 1 student student 0 Feb  4 18:48 film6.avi
-rw-rw-r-- 1 student student 0 Feb  4 18:48 snap5.jpg
-rw-rw-r-- 1 student student 0 Feb  4 18:48 snap6.jpg
-rw-rw-r-- 1 student student 0 Feb  4 18:48 song5.mp3
-rw-rw-r-- 1 student student 0 Feb  4 18:48 song6.mp3
```

- 8. Your project tasks are now complete, and it is time to clean up the projects.

Change to the **student** user's home directory. Attempt to delete both the **family** and **friends** project directories with a single **rmdir** command.

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

Using the **rmdir** command should fail because both subdirectories contain files.

- 9. Use the **rm -r** command to recursively delete both the **family** and **friends** subdirectories and their contents.

```
[student@servera ~]$ rm -r family friends
[student@servera ~]$ ls -l
total 0
drwxrwxr-x. 2 student student 108 Feb  4 18:36 Music
drwxrwxr-x. 2 student student 108 Feb  4 18:36 Pictures
drwxrwxr-x. 2 student student 108 Feb  4 18:36 Videos
drwxrwxr-x. 2 student student 108 Feb  4 18:48 work
```

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

```
[student@servera ~]$ cd work  
[student@servera work]$ rm song5.mp3 song6.mp3 snap5.jpg snap6.jpg \  
film5.avi film6.avi  
[student@servera work]$ ls -l  
total 0
```

- 11. Finally, from the **student** user's home directory, use the **rmdir** command to delete the **work** directory. The command should succeed now that it is empty.

```
[student@servera work]$ cd  
[student@servera ~]$ rmdir work  
[student@servera ~]$ ls -l  
total 0  
drwxrwxr-x. 2 student student 108 Feb 4 18:36 Music  
drwxrwxr-x. 2 student student 108 Feb 4 18:36 Pictures  
drwxrwxr-x. 2 student student 108 Feb 4 18:36 Videos
```

- 12. Exit from **servera**.

```
[student@servera ~]$ exit  
logout  
Connection to servera closed.  
[student@workstation ~]$
```

Finish

On **workstation**, run the **lab files-manage finish** script to finish this exercise. The script will remove all directories and files created during this exercise.

```
[student@workstation ~]$ lab files-manage finish
```

This concludes the guided exercise.

Making Links Between Files

Objectives

After completing this section, you should be able to make multiple file names reference the same file using hard links and symbolic (or "soft") links.

Managing Links Between Files

Hard Links and Soft Links

It is possible to create multiple names that point to the same file. There are two ways to do this: by creating a *hard link* to the file, or by creating a *soft link* (sometimes called a *symbolic link*) to the file. Each has its advantages and disadvantages.

Creating Hard Links

Every file starts with a single hard link, from its initial name to the data on the file system. When you create a new hard link to a file, you create another name that points to that same data. The new hard link acts exactly like the original file name. Once created, you cannot tell the difference between the new hard link and the original name of the file.

You can find out if a file has multiple hard links with the `ls -l` command. One of the things it reports is each file's *link count*, the number of hard links the file has.

```
[user@host ~]$ pwd
/home/user
[user@host ~]$ ls -l newfile.txt
-rw-r--r--. 1 user user 0 Mar 11 19:19 newfile.txt
```

In the preceding example, the link count of `newfile.txt` is 1. It has exactly one absolute path, which is `/home/user/newfile.txt`.

You can use the `ln` command to create a new hard link (another name) that points to an existing file. The command needs at least two arguments, a path to the existing file, and the path to the hard link that you want to create.

The following example creates a hard link named `newfile-link2.txt` for the existing file `newfile.txt` in the `/tmp` directory.

```
[user@host ~]$ ln newfile.txt /tmp/newfile-hlink2.txt
[user@host ~]$ ls -l newfile.txt /tmp/newfile-hlink2.txt
-rw-rw-r--. 2 user user 12 Mar 11 19:19 newfile.txt
-rw-rw-r--. 2 user user 12 Mar 11 19:19 /tmp/newfile-hlink2.txt
```

If you want to find out whether two files are hard links of each other, one way is to use the `-i` option with the `ls` command to list the files' *inode number*. If the files are on the same file system (discussed in a moment) and their inode numbers are the same, the files are hard links pointing to the same data.

```
[user@host ~]$ ls -il newfile.txt /tmp/newfile-hlink2.txt
8924107 -rw-rw-r--. 2 user user 12 Mar 11 19:19 newfile.txt
8924107 -rw-rw-r--. 2 user user 12 Mar 11 19:19 /tmp/newfile-hlink2.txt
```



Important

All hard links that reference the same file will have the same link count, access permissions, user and group ownerships, time stamps, and file content. If any of that information is changed using one hard link, all other hard links pointing to the same file will show the new information as well. This is because each hard link points to the same data on the storage device.

Even if the original file gets deleted, the contents of the file are still available as long as at least one hard link exists. Data is only deleted from storage when the last hard link is deleted.

```
[user@host ~]$ rm -f newfile.txt
[user@host ~]$ ls -l /tmp/newfile-hlink2.txt
-rw-rw-r--. 1 user user 12 Mar 11 19:19 /tmp/newfile-hlink2.txt
[user@host ~]$ cat /tmp/newfile-hlink2.txt
Hello World
```

Limitations of Hard Links

Hard links have some limitations. Firstly, hard links can only be used with regular files. You cannot use **ln** to create a hard link to a directory or special file.

Secondly, hard links can only be used if both files are on the same *file system*. The file-system hierarchy can be made up of multiple storage devices. Depending on the configuration of your system, when you change into a new directory, that directory and its contents may be stored on a different file system.

You can use the **df** command to list the directories that are on different file systems. For example, you might see output like the following:

```
[user@host ~]$ df
Filesystem      1K-blocks    Used Available Use% Mounted on
devtmpfs          886788      0   886788  0% /dev
tmpfs            902108      0   902108  0% /dev/shm
tmpfs            902108    8696   893412  1% /run
tmpfs            902108      0   902108  0% /sys/fs/cgroup
/dev/mapper/rhel_rhel8--root 10258432 1630460   8627972 16% /
/dev/sda1        1038336  167128   871208 17% /boot
tmpfs           180420      0   180420  0% /run/user/1000
[user@host ~]$
```

Files in two different "Mounted on" directories and their subdirectories are on different file systems. (The most specific match wins.) So, the system in this example, you can create a hard link between **/var/tmp/link1** and **/home/user/file** because they are both subdirectories of **/** but not any other directory on the list. But you cannot create a hard link between **/boot/test/badlink** and **/home/user/file** because the first file is in a subdirectory of **/boot** (on the "Mounted on" list) and the second file is not.

Creating Soft Links

The `ln -s` command creates a soft link, which is also called a "symbolic link." A soft link is not a regular file, but a special type of file that points to an existing file or directory.

Soft links have some advantages over hard links:

- They can link two files on different file systems.
- They can point to a directory or special file, not just a regular file.

In the following example, the `ln -s` command is used to create a new soft link for the existing file `/home/user/newfile-link2.txt` that will be named `/tmp/newfile-symlink.txt`.

```
[user@host ~]$ ln -s /home/user/newfile-link2.txt /tmp/newfile-symlink.txt
[user@host ~]$ ls -l newfile-link2.txt /tmp/newfile-symlink.txt
-rw-rw-r--. 1 user user 12 Mar 11 19:19 newfile-link2.txt
lrwxrwxrwx. 1 user user 11 Mar 11 20:59 /tmp/newfile-symlink.txt -> /home/user/
newfile-link2.txt
[user@host ~]$ cat /tmp/newfile-symlink.txt
Soft Hello World
```

In the preceding example, the first character of the long listing for `/tmp/newfile-symlink.txt` is `l` instead of `-`. This indicates that the file is a soft link and not a regular file. (A `d` would indicate that the file is a directory.)

When the original regular file gets deleted, the soft link will still point to the file but the target is gone. A soft link pointing to a missing file is called a "dangling soft link."

```
[user@host ~]$ rm -f newfile-link2.txt
[user@host ~]$ ls -l /tmp/newfile-symlink.txt
lrwxrwxrwx. 1 user user 11 Mar 11 20:59 /tmp/newfile-symlink.txt -> /home/user/
newfile-link2.txt
[user@host ~]$ cat /tmp/newfile-symlink.txt
cat: /tmp/newfile-symlink.txt: No such file or directory
```



Important

One side-effect of the dangling soft link in the preceding example is that if you later create a new file with the same name as the deleted file (`/home/user/newfile-link2.txt`), the soft link will no longer be "dangling" and will point to the new file.

Hard links do not work like this. If you delete a hard link and then use normal tools (rather than `ln`) to create a new file with the same name, the new file will not be linked to the old file.

One way to compare hard links and soft links that might help you understand how they work:

- A hard link points a name to data on a storage device
- A soft link points a name to another name, that points to data on a storage device

A soft link can point to a directory. The soft link then acts like a directory. Changing to the soft link with `cd` will make the current working directory the linked directory. Some tools may keep track of the fact that you followed a soft link to get there. For example, by default `cd` will update

your current working directory using the name of the soft link rather than the name of the actual directory. (There is an option, **-P**, that will update it with the name of the actual directory instead.)

In the following example, a soft link named **/home/user/configfiles** is created that points to the **/etc** directory.

```
[user@host ~]$ ln -s /etc /home/user/configfiles  
[user@host ~]$ cd /home/user/configfiles  
[user@host configfiles]$ pwd  
/home/user/configfiles
```



References

[ln\(1\) man page](#)

info ln ('ln': Make links between files)

► Guided Exercise

Making Links Between Files

In this exercise, you will create hard links and symbolic links and compare the results.

Outcomes

You should be able to create hard links and soft links between files.

Before You Begin

Log in as the **student** user on **workstation** using **student** as the password.

On **workstation**, run the **lab files-make start** command. This command runs a start script that determines if the **servera** host is reachable on the network and creates the files and working directories on **servera**.

```
[student@workstation ~]$ lab files-make start
```

- 1. Use the **ssh** command to log in to **servera** as the **student** user. The systems are configured to use SSH keys for authentication, and therefore a password is not required.

```
[student@workstation ~]$ ssh student@servera  
...output omitted...  
[student@servera ~]$
```

- 2. Create a hard link named **/home/student/backups/source.backup** for the existing file, **/home/student/files/source.file**.
- 2.1. View the link count for the file, **/home/student/files/source.file**.

```
[student@servera ~]$ ls -l files/source.file  
total 4  
-rw-r--r-- 1 student student 11 Mar 5 21:19 source.file
```

- 2.2. Create a hard link named **/home/student/backups/source.backup**. Link it to the file, **/home/student/files/source.file**.

```
[student@servera ~]$ ln /home/student/files/source.file \  
/home/student/backups/source.backup
```

- 2.3. Verify the link count for the original **/home/student/files/source.file** and the new linked file, **/home/student/backups/source.backup**. The link count should be **2** for both files.

```
[student@servera ~]$ ls -l /home/student/files/  
-rw-r--r--. 2 student student 11 Mar 5 21:19 source.file  
[student@servera ~]$ ls -l /home/student/backups/  
-rw-r--r--. 2 student student 11 Mar 5 21:19 source.backup
```

- 3. Create a soft link named **/home/student/tempdir** that points to the **/tmp** directory on **servera**.

- 3.1. Create a soft link named **/home/student/tempdir** and link it to **/tmp**.

```
[student@servera ~]$ ln -s /tmp /home/student/tempdir
```

- 3.2. Use the **ls -l** command to verify the newly created soft link.

```
[student@servera ~]$ ls -l /home/student/tempdir  
lrwxrwxrwx. 1 student student 4 Mar 5 22:04 /home/student/tempdir -> /tmp
```

- 4. Exit from **servera**.

```
[student@servera ~]$ exit  
logout  
Connection to servera closed.  
[student@workstation ~]$
```

Finish

On **workstation**, run the **lab files-make finish** script to finish this exercise. This script removes all files and directories created on **servera** during the exercise.

```
[student@workstation ~]$ lab files-make finish
```

This concludes the guided exercise.

Matching File Names with Shell Expansions

Objectives

After completing this section, you should be able to efficiently run commands affecting many files by using pattern matching features of the Bash shell.

Command-line Expansions

The Bash shell has multiple ways of expanding a command line including *pattern matching*, home directory expansion, string expansion, and variable substitution. Perhaps the most powerful of these is the path name-matching capability, historically called *globbing*. The Bash globbing feature, sometimes called “wildcards”, makes managing large numbers of files easier. Using metacharacters 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 metacharacters are replaced by the match list prior to command execution. Patterns that do not return matches display the original pattern request as literal text. The following are common metacharacters and pattern classes.

Table of Metacharacters and Matches

Pattern	Matches
*	Any string of zero or more characters.
?	Any single character.
[abc...]	Any one character in the enclosed class (between the square brackets).
[!abc...]	Any one character <i>not</i> in the enclosed class.
[^abc...]	Any one character <i>not</i> in the enclosed class.
[:alpha:]	Any alphabetic character.
[:lower:]	Any lowercase character.
[:upper:]	Any uppercase character.
[:alnum:]	Any alphabetic character or digit.
[:punct:]	Any printable character not a space or alphanumeric.
[:digit:]	Any single digit from 0 to 9.
[:space:]	Any single white space character. This may include tabs, newlines, carriage returns, form feeds, or spaces.

For the next few examples, pretend that you have run the following commands to create some sample files.

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

The first example will use simple pattern matches with the asterisk (*) and question mark (?) characters, and a class of characters, to match some of those file names.

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

Tilde Expansion

The tilde character (~), matches the current user's home directory. If it starts a string of characters other than a slash (/), the shell will interpret the string up to that slash as a user name, if one matches, and replace the string with the absolute path to that user's home directory. If no user name matches, then an actual tilde followed by the string of characters will be used instead.

In the following example the **echo** command is used to display the value of the tilde character. The **echo** command can also be used to display the values of brace and variable expansion characters, and others.

```
[user@host glob]$ echo ~root
/root
[user@host glob]$ echo ~user
/home/user
[user@host glob]$ echo ~/glob
/home/user/glob
[user@host 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. Also double-dot syntax (..) expands to a sequence such that **{m..p}** will expand to m n o p.

```
[user@host glob]$ echo {Sunday,Monday,Tuesday,Wednesday}.log
Sunday.log Monday.log Tuesday.log Wednesday.log
[user@host glob]$ echo file{1..3}.txt
```

```
file1.txt file2.txt file3.txt
[user@host glob]$ echo file{a..c}.txt
filea.txt fileb.txt filec.txt
[user@host glob]$ echo file{a,b}{1,2}.txt
filea1.txt filea2.txt fileb1.txt fileb2.txt
[user@host glob]$ echo file{a{1,2},b,c}.txt
filea1.txt filea2.txt fileb.txt filec.txt
[user@host glob]$
```

A practical use of brace expansion is to quickly create a number of files or directories.

```
[user@host glob]$ mkdir ../*RHEL{6,7,8}
[user@host glob]$ ls ../*RHEL*
RHEL6 RHEL7 RHEL8
[user@host glob]$
```

Variable Expansion

A variable acts like a named container that can store a value in memory. Variables make it easy to access and modify the stored data either from the command line or within a shell script.

You can assign data as a value to a variable using the following syntax:

```
[user@host ~]$ VARIABLENAME=value
```

You can use variable expansion to convert the variable name to its value on the command line. If a string starts with a dollar sign (\$), then the shell will try to use the rest of that string as a variable name and replace it with whatever value the variable has.

```
[user@host ~]$ USERNAME=operator
[user@host ~]$ echo $USERNAME
operator
```

To help avoid mistakes due to other shell expansions, you can put the name of the variable in curly braces, for example `$(VARIABLENAME)`.

```
[user@host ~]$ USERNAME=operator
[user@host ~]$ echo ${USERNAME}
operator
```

Shell variables and ways to use them will be covered in more depth later in this course.

Command Substitution

Command substitution allows the output of a command to replace the command itself on the command line. Command substitution occurs when a command is enclosed in parentheses, and preceded by a dollar sign (\$). The `$(command)` form can nest multiple command expansions inside each other.

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

**Note**

An older form of command substitution uses backticks: ` **command** `. Disadvantages to the backticks form include: 1) it can be easy to visually confuse backticks with single quote marks, and 2) backticks cannot be nested.

Protecting Arguments from Expansion

Many characters have special meaning in the Bash shell. To keep the shell from performing shell expansions on parts of your command line, you can *quote* and *escape* characters and strings.

The backslash (\) is an escape character in the Bash shell. It will protect the character immediately following it from expansion.

```
[user@host glob]$ echo The value of $HOME is your home directory.
The value of /home/user is your home directory.
[user@host glob]$ echo The value of \$HOME is your home directory.
The value of $HOME is your home directory.
[user@host glob]$
```

In the preceding example, protecting the dollar sign from expansion caused Bash to treat it as a regular character and it did not perform variable expansion on **\$HOME**.

To protect longer character strings, single quotes (') or double quotes ("") are used to enclose strings. They have slightly different effects. Single quotes stop all shell expansion. Double quotes stop *most* shell expansion.

Use double quotation marks to suppress globbing and shell expansion, but still allow command and variable substitution.

```
[user@host glob]$ myhost=$(hostname -s); echo $myhost
host
[user@host glob]$ echo "***** hostname is ${myhost} *****"
***** hostname is host *****
[user@host glob]$
```

Use single quotation marks to interpret *all* text literally.

```
[user@host glob]$ echo "Will variable $myhost evaluate to $(hostname -s)?"
Will variable host evaluate to host?
[user@host glob]$ echo 'Will variable $myhost evaluate to $(hostname -s)?'
Will variable $myhost evaluate to $(hostname -s)?
[user@host glob]$
```



Important

The single quote (') and the command substitution backtick (`) can be easy to confuse, both on the screen and on the keyboard. Using one when you mean to use the other will lead to unexpected shell behavior.



References

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

► Quiz

Matching File Names with Shell Expansions

Choose the correct answers to the following questions:

► 1. Which pattern will match only filenames ending with "b"?

- a. **b***
- b. ***b**
- c. ***b***
- d. **[!b]***

► 2. Which pattern will match only filenames beginning with "b"?

- a. **b***
- b. ***b**
- c. ***b***
- d. **[!b]***

► 3. Which pattern will match only filenames where the first character is not "b"?

- a. **b***
- b. ***b**
- c. ***b***
- d. **[!b]***

► 4. Which pattern will match all filenames containing a "b"?

- a. **b***
- b. ***b**
- c. ***b***
- d. **[!b]***

► 5. Which pattern will match only filenames that contain a number?

- a. ***#***
- b. ***[[:digit:]]***
- c. ***[digit]***
- d. **[0-9]**

► **6. Which pattern will match only filenames that begin with an uppercase letter?**

- a. ^?*
- b. ^*
- c. [upper]*
- d. [[:upper:]]*
- e. [[CAP]]*

► **7. Which pattern will match only filenames at least three characters in length?**

- a. ???*
- b. ???
- c. \3*
- d. +++*
- e. ...*

► Solution

Matching File Names with Shell Expansions

Choose the correct answers to the following questions:

► 1. Which pattern will match only filenames ending with "b"?

- a. b*
- b. *b
- c. *b*
- d. [!b]*

► 2. Which pattern will match only filenames beginning with "b"?

- a. b*
- b. *b
- c. *b*
- d. [!b]*

► 3. Which pattern will match only filenames where the first character is not "b"?

- a. b*
- b. *b
- c. *b*
- d. [!b]*

► 4. Which pattern will match all filenames containing a "b"?

- a. b*
- b. *b
- c. *b*
- d. [!b]*

► 5. Which pattern will match only filenames that contain a number?

- a. *#*
- b. *[[:digit:]]*
- c. *[digit]*
- d. [0-9]

► **6. Which pattern will match only filenames that begin with an uppercase letter?**

- a. `^?*`
- b. `^*`
- c. `[upper]*`
- d. `[:upper:]*`
- e. `[[CAP]]*`

► **7. Which pattern will match only filenames at least three characters in length?**

- a. `???*`
- b. `???`
- c. `\3*`
- d. `+++*`
- e. `...*`

▶ Lab

Managing Files from the Command Line

Performance Checklist

In this lab, you will efficiently create, move, and remove files and directories by using the shell and a variety of file name matching techniques.

Outcomes

You should be able to:

- Use wildcards to locate and manipulate files.

Before You Begin

Log in to **workstation** as **student** using **student** as the password.

On **workstation**, run the **lab files-review start** command. The command runs a start script that determines if the **serverb** machine is reachable on the network.

```
[student@workstation ~]$ lab files-review start
```

1. Use the **ssh** command to log in to **serverb** as the **student** user. The systems are configured to use SSH keys for authentication, and therefore a password is not required.
2. Before you create project files, use the **mkdir** command with brace expansion to create empty project planning documents in the **/home/student/Documents/project_plans** directory. (Hint: if **~/Documents** does not exist, the **-p** option for the **mkdir** command will create it.)
Create two empty files in the **~/Documents/project_plans** directory:
season1_project_plan.odf and **season2_project_plan.odf**.
3. Create sets of empty practice files to use in this lab. If you do not immediately recognize the intended shell expansion shortcut, 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.
4. 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.
5. Use a single command to create two subdirectories named **season1** and **season2** under the **Videos** directory, to organize the TV episodes.
6. Move the appropriate TV episodes into the season subdirectories. Use only two commands, specifying destinations using relative syntax.
7. Create a 2-level directory hierarchy with a single command to organize the mystery book chapters. Create **my_bestseller** under the **Documents** directory, and **chapters** under the new **my_bestseller** directory.

8. Create three more subdirectories directly under the **my_bestseller** directory using a single command. Name these subdirectories **editor**, **changes**, and **vacation**. The **-p** option (create parents) is not needed because the **my_bestseller** parent directory already exists.
9. Change to the **chapters** directory. Using the tilde (~) home directory shortcut to specify the source files, move all book chapters to the **chapters** directory, which is now your current directory. What is the simplest syntax to specify the destination directory?
10. You sent the first two chapters to the editor for review. Move only those two chapters to the **editor** directory to avoid modifying them during the review. Starting from the **chapters** subdirectory, use brace expansion with a range to specify the chapter file names to move and a relative path for the destination directory.
11. While on vacation you intend to write chapters 7 and 8. Use a single command to move the files from the **chapters** directory to the **vacation** directory. Specify the chapter file names using brace expansion with a list of strings and without using wildcard characters.
12. Change your working directory to **~/Videos/season2**, and then copy the first episode of the season to the **vacation** directory.
13. Use a single **cd** command to change from your working directory to the **~/Documents/my_bestseller/vacation** directory. List its files. Use the *previous working directory* argument to return to the **season2** directory. (This will succeed if the last directory change with the **cd** command was accomplished with one command rather than several **cd** commands.) From the **season2** directory, copy the episode 2 file into the **vacation** directory. Use the shortcut again to return to the **vacation** directory.
14. The authors of chapters 5 and 6 want to experiment with possible changes. Copy both files from the **~/Documents/my_bestseller/chapters** directory to the **~/Documents/my_bestseller/changes** directory to prevent these changes from modifying original files. Navigate to the **~/Documents/my_bestseller** directory. Use square-bracket pattern matching to specify which chapter numbers to match in the filename argument of the **cp** command.
15. Change your current directory to the **changes** directory.
Use the **date +%F** command with command substitution to copy **mystery_chapter5.odf** to a new file which includes the full date. The name should have the form **mystery_chapter5_YYYY-MM-DD.odf**.
Make another copy of **mystery_chapter5.odf**, appending the current time stamp (as the number of seconds since the epoch, 1970-01-01 00:00 UTC) to ensure a unique file name. Use command substitution with the **date +%s** command to accomplish this.
16. After further review, you decide that the plot changes are not necessary. Delete the **changes** directory.
If necessary, navigate to the **changes** directory and delete all the files within the directory. You cannot delete a directory while it is the current working directory. Change to the parent directory of the **changes** directory. Try to delete the empty directory using the **rm** command without the **-r** recursive option. This attempt should fail. Finally, use the **rmdir** command to delete the empty directory, which will succeed.
17. 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 **student** user's home directory.
18. Create a hard link to the **~/Documents/project_plans/season2_project_plan.odf** file named **~/Documents/backups/season2_project_plan.odf.back**. A hard link will protect against accidental deletion of the original file and will keep the backup file updated as changes are made to the original.

19. Exit from **serverb.**

```
[student@serverb ~]$ exit  
logout  
Connection to serverb closed.  
[student@workstation ~]$
```

Evaluation

On **workstation**, run the **lab files-review grade** script to confirm success on this lab.

```
[student@workstation ~]$ lab files-review grade
```

Finish

On **workstation**, run the **lab files-review finish** script to finish this lab. This script removes all files and directories created on **serverb** during the lab exercise.

```
[student@workstation ~]$ lab files-review finish
```

This concludes the lab.

► Solution

Managing Files from the Command Line

Performance Checklist

In this lab, you will efficiently create, move, and remove files and directories by using the shell and a variety of file name matching techniques.

Outcomes

You should be able to:

- Use wildcards to locate and manipulate files.

Before You Begin

Log in to **workstation** as **student** using **student** as the password.

On **workstation**, run the **lab files-review start** command. The command runs a start script that determines if the **serverb** machine is reachable on the network.

```
[student@workstation ~]$ lab files-review start
```

1. Use the **ssh** command to log in to **serverb** as the **student** user. The systems are configured to use SSH keys for authentication, and therefore a password is not required.

```
[student@workstation ~]$ ssh student@serverb
...output omitted...
[student@serverb ~]$
```

2. Before you create project files, use the **mkdir** command with brace expansion to create empty project planning documents in the **/home/student/Documents/project_plans** directory. (Hint: if **~/Documents** does not exist, the **-p** option for the **mkdir** command will create it.)

Create two empty files in the **~/Documents/project_plans** directory: **season1_project_plan.odf** and **season2_project_plan.odf**.

```
[student@serverb ~]$ mkdir -p ~/Documents/project_plans
[student@serverb ~]$ touch \
~/Documents/project_plans/{season1,season2}_project_plan.odf
[student@serverb ~]$ ls -lR Documents/
Documents:/
total 0
drwxrwxr-x. 2 student student 70 Jan 31 18:20 project_plans

Documents/project_plans:
total 0
-rw-rw-r--. 1 student student 0 Jan 31 18:20 season1_project_plan.odf
-rw-rw-r--. 1 student student 0 Jan 31 18:20 season2_project_plan.odf
```

3. Create sets of empty practice files to use in this lab. If you do not immediately recognize the intended shell expansion shortcut, 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@serverb ~]$ touch tv_season{1..2}_episode{1..6}.ogg
[student@serverb ~]$ ls tv*
tv_season1_episode1.ogg  tv_season1_episode5.ogg  tv_season2_episode3.ogg
tv_season1_episode2.ogg  tv_season1_episode6.ogg  tv_season2_episode4.ogg
tv_season1_episode3.ogg  tv_season2_episode1.ogg  tv_season2_episode5.ogg
tv_season1_episode4.ogg  tv_season2_episode2.ogg  tv_season2_episode6.ogg
```

4. 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@serverb ~]$ touch mystery_chapter{1..8}.odf
[student@serverb ~]$ ls mys*
mystery_chapter1.odf  mystery_chapter4.odf  mystery_chapter7.odf
mystery_chapter2.odf  mystery_chapter5.odf  mystery_chapter8.odf
mystery_chapter3.odf  mystery_chapter6.odf
```

5. Use a single command to create two subdirectories named **season1** and **season2** under the **Videos** directory, to organize the TV episodes.

```
[student@serverb ~]$ mkdir -p Videos/season{1..2}
[student@serverb ~]$ ls Videos
season1  season2
```

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

```
[student@serverb ~]$ mv tv_season1* Videos/season1
[student@serverb ~]$ mv tv_season2* Videos/season2
[student@serverb ~]$ ls -R Videos
Videos:
season1  season2

Videos/season1:
tv_season1_episode1.ogg  tv_season1_episode3.ogg  tv_season1_episode5.ogg
tv_season1_episode2.ogg  tv_season1_episode4.ogg  tv_season1_episode6.ogg

Videos/season2:
tv_season2_episode1.ogg  tv_season2_episode3.ogg  tv_season2_episode5.ogg
tv_season2_episode2.ogg  tv_season2_episode4.ogg  tv_season2_episode6.ogg
```

7. Create a 2-level directory hierarchy with a single command to organize the mystery book chapters. Create **my_bestseller** under the **Documents** directory, and **chapters** under the new **my_bestseller** directory.

```
[student@serverb ~]$ mkdir -p Documents/my_bestseller/chapters
[student@serverb ~]$ ls -R Documents
Documents:
my_bestseller project_plans

Documents/my_bestseller:
chapters

Documents/my_bestseller/chapters:
Documents/project_plans:
season1_project_plan.odf  season2_project_plan.odf
```

8. Create three more subdirectories directly under the **my_bestseller** directory using a single command. Name these subdirectories **editor**, **changes**, and **vacation**. The **-p** option (create parents) is not needed because the **my_bestseller** parent directory already exists.

```
[student@serverb ~]$ mkdir Documents/my_bestseller/{editor,changes,vacation}
[student@serverb ~]$ ls -R Documents
Documents:
my_bestseller project_plans

Documents/my_bestseller:
changes chapters editor vacation

Documents/my_bestseller/changes:

Documents/my_bestseller/chapters:

Documents/my_bestseller/editor:

Documents/my_bestseller/vacation:

Documents/project_plans:
season1_project_plan.odf  season2_project_plan.odf
```

9. Change to the **chapters** directory. Using the tilde (~) home directory shortcut to specify the source files, move all book chapters to the **chapters** directory, which is now your current directory. What is the simplest syntax to specify the destination directory?

```
[student@serverb ~]$ cd Documents/my_bestseller/chapters
[student@serverb chapters]$ mv ~/mystery_chapter* .
[student@serverb chapters]$ ls
mystery_chapter1.odf mystery_chapter4.odf mystery_chapter7.odf
mystery_chapter2.odf mystery_chapter5.odf mystery_chapter8.odf
mystery_chapter3.odf mystery_chapter6.odf
```

10. You sent the first two chapters to the editor for review. Move only those two chapters to the **editor** directory to avoid modifying them during the review. Starting from the chapters

subcategory, use brace expansion with a range to specify the chapter file names to move and a relative path for the destination directory.

```
[student@serverb chapters]$ mv mystery_chapter{1..2}.odf ../editor
[student@serverb chapters]$ ls
mystery_chapter3.odf mystery_chapter5.odf mystery_chapter7.odf
mystery_chapter4.odf mystery_chapter6.odf mystery_chapter8.odf
[student@serverb chapters]$ ls ../editor
mystery_chapter1.odf mystery_chapter2.odf
```

- While on vacation you intend to write chapters 7 and 8. Use a single command to move the files from the **chapters** directory to the **vacation** directory. Specify the chapter file names using brace expansion with a list of strings and without using wildcard characters.

```
[student@serverb chapters]$ mv mystery_chapter{7,8}.odf ../vacation
[student@serverb chapters]$ ls
mystery_chapter3.odf mystery_chapter5.odf
mystery_chapter4.odf mystery_chapter6.odf
[student@serverb chapters]$ ls ../vacation
mystery_chapter7.odf mystery_chapter8.odf
```

- Change your working directory to **~/Videos/season2**, and then copy the first episode of the season to the **vacation** directory.

```
[student@serverb chapters]$ cd ~/Videos/season2
[student@serverb season2]$ cp *episode1.ogg ~/Documents/my_bestseller/vacation
```

- Use a single **cd** command to change from your working directory to the **~/Documents/my_bestseller/vacation** directory. List its files. Use the *previous working directory* argument to return to the **season2** directory. (This will succeed if the last directory change with the **cd** command was accomplished with one command rather than several **cd** commands.) From the **season2** directory, copy the episode 2 file into the **vacation** directory. Use the shortcut again to return to the **vacation** directory.

```
[student@serverb season2]$ cd ~/Documents/my_bestseller/vacation
[student@serverb vacation]$ ls
mystery_chapter7.odf mystery_chapter8.odf tv_season2_episode1.ogg
[student@serverb vacation]$ cd -
/home/ec2-user/Videos/season2
[student@serverb season2]$ cp *episode2.ogg ~/Documents/my_bestseller/vacation
[student@serverb vacation]$ cd -
/home/ec2-user/Documents/my_bestseller/vacation
[student@serverb vacation]$ ls
mystery_chapter7.odf tv_season2_episode1.ogg
mystery_chapter8.odf tv_season2_episode2.ogg
```

- The authors of chapters 5 and 6 want to experiment with possible changes. Copy both files from the **~/Documents/my_bestseller/chapters** directory to the **~/Documents/my_bestseller/changes** directory to prevent these changes from modifying original files. Navigate to the **~/Documents/my_bestseller** directory. Use square-bracket pattern

matching to specify which chapter numbers to match in the filename argument of the **cp** command.

```
[student@serverb vacation]$ cd ~/Documents/my_bestseller
[student@serverb my_bestseller]$ cp chapters/mystery_chapter[56].odf changes
[student@serverb my_bestseller]$ ls chapters
mystery_chapter3.odf mystery_chapter5.odf
mystery_chapter4.odf mystery_chapter6.odf
[student@serverb my_bestseller]$ ls changes
mystery_chapter5.odf mystery_chapter6.odf
```

15. Change your current directory to the **changes** directory.

Use the **date +%F** command with command substitution to copy **mystery_chapter5.odf** to a new file which includes the full date. The name should have the form **mystery_chapter5_YYYY-MM-DD.odf**.

Make another copy of **mystery_chapter5.odf**, appending the current time stamp (as the number of seconds since the epoch, 1970-01-01 00:00 UTC) to ensure a unique file name. Use command substitution with the **date +%s** command to accomplish this.

```
[student@serverb my_bestseller]$ cd changes
[student@serverb changes]$ cp mystery_chapter5.odf \
mystery_chapter5_$(date +%F).odf
[student@serverb changes]$ cp mystery_chapter5.odf \
mystery_chapter5_$(date +%s).odf
[student@serverb changes]$ ls
mystery_chapter5_1492545076.odf mystery_chapter5.odf
mystery_chapter5_2017-04-18.odf mystery_chapter6.odf
```

16. After further review, you decide that the plot changes are not necessary. Delete the **changes** directory.

If necessary, navigate to the **changes** directory and delete all the files within the directory. You cannot delete a directory while it is the current working directory. Change to the parent directory of the **changes** directory. Try to delete the empty directory using the **rm** command without the **-r** recursive option. This attempt should fail. Finally, use the **rmdir** command to delete the empty directory, which will succeed.

```
[student@serverb changes]$ rm mystery*
[student@serverb changes]$ cd ..
[student@serverb my_bestseller]$ rm changes
rm: cannot remove 'changes': Is a directory
[student@serverb my_bestseller]$ rmdir changes
[student@serverb my_bestseller]$ ls
chapters editor vacation
```

17. 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 **student** user's home directory.

```
[student@serverb my_bestseller]$ rm -r vacation
[student@serverb my_bestseller]$ ls
chapters editor
[student@serverb my_bestseller]$ cd
[student@serverb ~]$
```

18. Create a hard link to the **~/Documents/project_plans/season2_project_plan.odf** file named **~/Documents/backups/season2_project_plan.odf.back**. A hard link will protect against accidental deletion of the original file and will keep the backup file updated as changes are made to the original.

Notice that the link count is **2** for both **season2_project_plan.odf.back** and **season2_project_plan.odf** files.

```
[student@serverb ~]$ mkdir ~/Documents/backups
[student@serverb ~]$ ln ~/Documents/project_plans/season2_project_plan.odf \
~/Documents/backups/season2_project_plan.odf.back
[student@serverb ~]$ ls -lR ~/Documents/
/home/student/Documents/:
total 0
drwxrwxr-x. 2 student student 43 Jan 31 18:59 backups
drwxrwxr-x. 4 student student 36 Jan 31 19:42 my_bestseller
drwxrwxr-x. 2 student student 70 Jan 31 18:20 project_plans

/home/student/Documents/backups:
total 4
-rw-rw-r--. 2 student student 0 Jan 31 19:05 season2_project_plan.odf.back

/home/student/Documents/my_bestseller:
total 0
drwxrwxr-x. 2 student student 118 Jan 31 19:39 chapters
drwxrwxr-x. 2 student student 62 Jan 31 19:38 editor

/home/student/Documents/my_bestseller/chapters:
total 0
-rw-rw-r--. 1 student student 0 Jan 31 19:18 mystery_chapter3.odf
-rw-rw-r--. 1 student student 0 Jan 31 19:18 mystery_chapter4.odf
-rw-rw-r--. 1 student student 0 Jan 31 19:18 mystery_chapter5.odf
-rw-rw-r--. 1 student student 0 Jan 31 19:18 mystery_chapter6.odf

/home/student/Documents/my_bestseller/editor:
total 0
-rw-rw-r--. 1 student student 0 Jan 31 19:18 mystery_chapter1.odf
-rw-rw-r--. 1 student student 0 Jan 31 19:18 mystery_chapter2.odf

/home/student/Documents/project_plans:
total 4
-rw-rw-r--. 1 student student 0 Jan 31 18:20 season1_project_plan.odf
-rw-rw-r--. 2 student student 0 Jan 31 19:05 season2_project_plan.odf
```

19. Exit from **serverb**.

```
[student@serverb ~]$ exit  
logout  
Connection to serverb closed.  
[student@workstation ~]$
```

Evaluation

On **workstation**, run the **lab files-review grade** script to confirm success on this lab.

```
[student@workstation ~]$ lab files-review grade
```

Finish

On **workstation**, run the **lab files-review finish** script to finish this lab. This script removes all files and directories created on **serverb** during the lab exercise.

```
[student@workstation ~]$ lab files-review finish
```

This concludes the lab.

Summary

In this chapter, you learned:

- Files on a Linux system are organized into a single inverted tree of directories, known as a file-system hierarchy.
- Absolute paths start with a / and specify the location of a file in the file-system hierarchy.
- Relative paths do not start with a / and specify the location of a file relative to the current working directory.
- Five key commands are used to manage files: **mkdir**, **rmdir**, **cp**, **mv**, and **rm**.
- Hard links and soft links are different ways to have multiple file names point to the same data.
- The Bash shell provides pattern matching, expansion, and substitution features to help you efficiently run commands.

Chapter 4

Getting Help in Red Hat Enterprise Linux

Goal

Resolve problems by using local help systems.

Objectives

- Find information in local Linux system manual pages.
- Find information from local documentation in GNU Info.

Sections

- Reading Manual Pages (and Guided Exercise)
- Reading Info Documentation (and Guided Exercise)

Lab

Getting Help in Red Hat Enterprise Linux

Reading Manual Pages

Objectives

After completing this section, you will be able to find information in local Linux system manual pages.

Introducing the `man` command

One source of documentation that is generally available on the local system are system manual pages or *man pages*. These pages are shipped as part of the software packages for which they provide documentation, and can be accessed from the command line by using the `man` command.

The historical Linux Programmer's Manual, from which man pages originate, was large enough to be multiple printed sections. Each section contains information about a particular topic.

Common 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>)

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`. Contents are displayed one screen at a time. The `man` command searches manual sections in alphanumeric order. 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)`.

Navigate and Search Man Pages

The ability to efficiently search for topics and navigate man pages is a critical administration skill. GUI tools make it easy to configure common system resources, but using the command-line

interface is still more efficient. To effectively navigate the command line, you must be able to find the information you need in man pages.

The following table lists basic navigation commands when viewing man pages:

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 backward (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
Shift+N	Repeat previous search backward (up) in the man page
G	Go to start of the man page.
Shift+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 that 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.

Reading Man Pages

Each topic is separated into several parts. Most topics share the same headings and are presented in the same order. Typically a topic does not feature all headings, because not all headings apply for all topics.

Common headings are:

Headings

Heading	Description
NAME	Subject name. Usually a command or file name. Very brief description.
SYNOPSIS	Summary of the command syntax.
DESCRIPTION	In-depth description to provide a basic understanding of the topic.
OPTIONS	Explanation of the command execution options.
EXAMPLES	Examples of how to use the command, function, or file.
FILES	A list of files and directories related to the man page.
SEE ALSO	Related information, normally other man page topics.
BUGS	Known bugs in the software.
AUTHOR	Information about who has contributed to the development of the topic.

Searching for man pages by keyword

A keyword search of man pages is performed with `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 generally 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** (uppercase) option performs a full-text page search, not just titles and descriptions like the **-k** option. A full-text search uses greater system 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

► Guided Exercise

Reading Manual Pages

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

Outcomes

You should be able to use the **man** Linux manual system and find useful information by searching and browsing.

Before You Begin

Log in to **workstation** as **student** using **student** as the password.

On **workstation**, run the **lab help-manual start** command. It creates a file called **manual**.

```
[student@workstation ~]$ lab help-manual start
```

- 1. On **workstation** view the **gedit** man page. View the options for editing a specific file using **gedit** from the command line.

Use one of the options from the **gedit** man page to open the **/home/student/manual** file using **gedit** with the cursor at the end of the file.

- 1.1. View the **gedit** man page.

```
[student@workstation ~]$ man gedit
```

```
GEDIT(1)      General Commands Manual      GEDIT(1)
NAME
       gedit - text editor for the GNOME Desktop

SYNOPSIS
       gedit [OPTION...] [FILE...] [+LINE[:COLUMN]]
       gedit [OPTION...] -
...output omitted...
```

- 1.2. In the **gedit** man page, learn the options for editing a specific file from the command line.

```
...output omitted...
FILE Specifies the file to open when gedit starts.
...output omitted...
+LINE For the first file, go to the line specified by LINE (do not insert
a space between the "+" sign and the number). If LINE is missing, go to the last
line.
...output omitted...
```

Press **q** to quit the man page.

13. Use the **gedit +** command to open the **manual** file. The missing line number next to **+** option opens a file passed as an argument with cursor at the end of the last line.

```
[student@workstation ~]$ gedit + manual
```

```
the quick brown fox just came over to greet the lazy poodle!
```

Confirm that the file is opened with the cursor at the end of the last line in the file.
Press **Ctrl+q** to close the application.

▶ 2. Read the **su(1)** man page.

Note that when the *user* is omitted the **su** command assumes the user is **root**. If the **su** command is followed by a single dash (-), it starts a child login shell. Without the dash, a non-login child shell is created that matches the user's current environment.

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

```
SU(1) User Commands SU(1)
NAME
    su - run a command with substitute user and group ID

SYNOPSIS
    su [options] [-] [user [argument...]]

DESCRIPTION
    su allows to run commands with a substitute user and group ID.
    When called without arguments, su defaults to running an interactive
    shell as root.
...output omitted...
OPTIONS
...output omitted...
-, -l, --login
    Start the shell as a login shell with an environment similar to a real login
...output omitted...
```



Note

Note that comma-separated options on a single line, such as **-**, **-l**, and **--login**, all result in the same behavior.

Press **q** to quit the man page.

- 3. The **man** command also has its own manual pages.

```
[student@workstation ~]$ man man
MAN(1)           Manual pager utils                               MAN(1)

NAME
    man - an interface to the on-line reference manuals
...output omitted...

DESCRIPTION
    man is the system's manual pager. Each page argument given to man is
    normally the name of a program, utility or function. The manual page
    associated with each of these arguments is then found and displayed.
    A section, if provided, will direct man to look only in that section
    of the manual.
...output omitted...
```

Press **q** to quit the man page.

- 4. All man pages are located in **/usr/share/man**. Locate the binary, source, and manual pages located in the **/usr/share/man** directory by using the **whereis** command.

```
[student@workstation ~]$ whereis passwd
passwd: /usr/bin/passwd /etc/passwd /usr/share/man/man1/passwd.1.gz /usr/share/
man/man5/passwd.5.gz
```

- 5. Use the **man -k zip** command to list detailed information about a ZIP archive.

```
[student@workstation ~]$ man -k zip
...output omitted...
zipinfo (1)      - list detailed information about a ZIP archive
zipnote (1)      - write the comments in zipfile to stdout, edit comments and
    rename files in zipfile
zipsplit (1)     - split a zipfile into smaller zipfiles
```

- 6. Use the **man -k boot** to list the man page containing a list of parameters that can be passed to the kernel at boot time.

```
[student@workstation ~]$ man -k boot
...output omitted...
bootctl (1)      - Control the firmware and boot manager settings
bootparam (7)     - introduction to boot time parameters of the Linux kernel
bootup (7)       - System bootup process
...output omitted...
```

- 7. Use the **man -k ext4** to find the command used to tune ext4 file system parameters.

```
[student@workstation ~]$ man -k ext4
...output omitted...
resize2fs (8)           - ext2/ext3/ext4 file system resizer
tune2fs (8)           - adjust tunable filesystem parameters on ext2/ext3/ext4
filesystems
```

Finish

On **workstation**, run the **lab help-manual finish** script to complete this exercise.

```
[student@workstation ~]$ lab help-manual finish
```

This concludes the guided exercise.

Reading Info Documentation

Objectives

After completing this section, students should be able to find information from local documentation in GNU Info.

Introducing GNU Info

Man pages have a 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.



Important

You might wonder why there are two local documentation systems, man pages and Info documents. Some of the reasons for this are practical in nature, and some have to do with the way Linux and its applications have been developed by various open source communities over the years.

Man pages have a much more formal format, and typically document a specific command or function from a software package, and are structured as individual text files. Info documents typically cover particular software packages as a whole, tend to have more practical examples of how to use the software, and are structured as hypertext documents.

You should be familiar with both systems in order to take maximum advantage of the information available to you from the system.

Reading Info Documentation

To launch the Info document viewer, use the `pinfo` command. `pinfo` opens in the *top directory*.

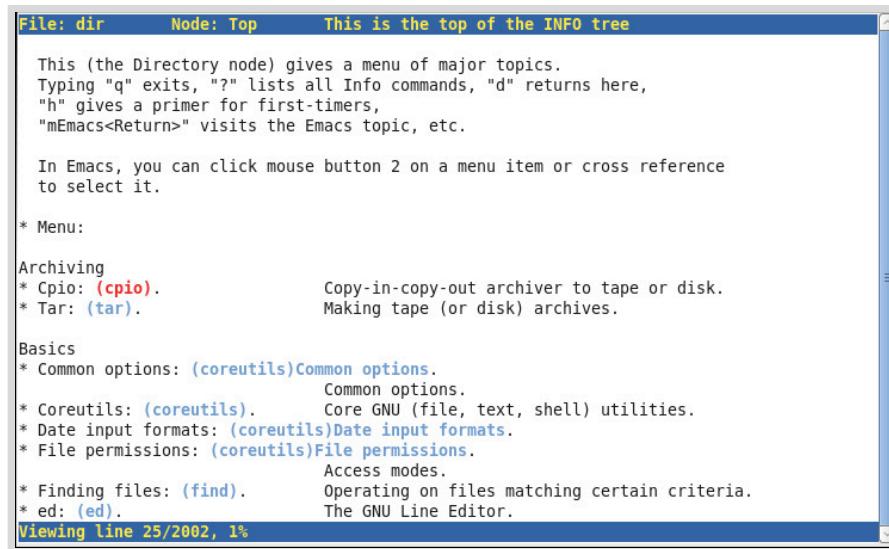


Figure 4.1: pinfo Info document viewer, top directory

Info documentation is comprehensive and hyperlinked. It is possible to output info pages to multiple formats. By contrast, man pages are optimized for printed output. The Info 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 the **pinfo** command.

A typical man page has a small amount of content focusing on one particular topic, command, tool, or file. The Info documentation is a comprehensive document. Info provides the following improvements:

- One single document for a large system containing all the necessary information for that system
- Hyperlinks
- A complete browsable document index
- A full text search of the entire document

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

```
[user@host ~]$ man tar
[user@host ~]$ pinfo tar
```

The **pinfo** reader is more advanced than the original **info** command. To browse a specific topic, use the **pinfo topic** command. The **pinfo** command without an argument opens the top directory. New documentation becomes available in **pinfo** when their software packages are installed.



Note

If no Info topic exists in the system for a particular entry that you requested, Info will look for a matching man page and display that instead.

Comparing GNU Info and Man Page Navigation

The **pinfo** command and the **man** command use slightly different navigational keystrokes. The following table compares the navigational keystrokes for both commands:

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	g
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*)

pinfo pinfo (*Documentation for pinfo*)

The GNU Project

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

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

► Guided Exercise

Reading Info Documentation

In this exercise, you will look up information stored in GNU Info documents by navigating those documents with command-line tools.

Outcomes

You should be able to navigate GNU Info documentation with command-line tools.

Before You Begin

Log in to **workstation** as **student** using **student** as the password.

On **workstation**, run the **lab help-info start** command.

```
[student@workstation ~]$ lab help-info start
```

- 1. On **workstation** launch **pinfo** without any arguments.

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

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

Use **UpArrow** or **DownArrow** until **(coreutils) Common options** is highlighted.

Basics

* Bash: (bash)

The GNU Bourne-Again SHell.

* Common options: (coreutils)Common options

Figure 4.2: Bash documentation

- 3. Press **Enter** to view this topic

File: coreutils.info, Node: Common options, Next: Output of entire files, Prev: Introduction, Up: Top

2 Common options

Certain options are available in all of these programs. Rather than writing identical descriptions for each of the programs, they are described here. (In fact, every GNU program accepts (or should accept) these options.)

Normally options and operands can appear in any order, and programs act as if all the options appear before any operands. For example, 'sort -r passwd -t :' acts like 'sort -r -t passwd', since ':' is an option-argument of '-t'. However, if the 'POSIXLY_CORRECT' environment variable is set, options must appear before operands, unless otherwise specified for a particular command.

Figure 4.3: Common options info topics

- 4. Browse through this Info topic. Learn whether long-style options can be abbreviated.

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

- ▶ 5. 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.

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

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

- ▶ 7. Return to the top level 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 moves directly to the directory of topics.

- ▶ 8. Search for the pattern **coreutils** and select that topic.

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

```
* Coreutils: (coreutils).          Core GNU (file, text, shell) utilities.
```

Figure 4.4: Search result

- ▶ 9. In the menu at the top, locate and select **Output of entire files** by pressing **n**.
Browse the topic.

Use **Enter** to select **cat invocation**. Use the arrow keys to browse the topic.

- ▶ 10. Move up two levels to return to **GNU Coreutils**. Move to **Summarizing files**.

Press **Enter** to select the topic then browse the topic.

- ▶ 11. Press **q** to quit **pinfo**.

- ▶ 12. Use the **pinfo** command again, specifying **coreutils** as the destination topic from the command line.

```
[student@workstation ~]$ pinfo coreutils
```

- ▶ 13. Select the **Disk usage** topic.

Press **DownArrow** to highlight **Disk usage**, then press **Enter** to select this topic.

- ▶ 14. Read the **df invocation** and **du invocation** 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 when you are finished.

Finish

On **workstation**, run the **lab help-info finish** script to complete this exercise.

```
[student@workstation ~]$ lab help-info finish
```

This concludes the guided exercise.

▶ Lab

Getting Help in Red Hat Enterprise Linux

Performance Checklist

In this lab, you will look up information to help you complete tasks in man pages and GNU Info documents.

Outcomes

You should be able to:

- Locate relevant commands by searching man pages and Info nodes.
- Learn new options for commonly used documentation commands.
- Use appropriate tools to view and print documentation and other non-text formatted files.

Before You Begin

Log in to **workstation** as **student** using **student** as the password.

On **workstation**, run the **lab help-review start** command.

```
[student@workstation ~]$ lab help-review start
```

1. On **workstation**, determine how to prepare a man page for printing. Specifically, find what format or rendering language is used for printing.
2. Create a formatted output file of the **passwd** man page. Call the file **passwd.ps**. Determine the file content format. Inspect the contents of the **passwd.ps** file.



Note

Create formatted output of the **passwd** man page using the following command:

```
[student@workstation ~]$ man -t passwd > passwd.ps
```

The **>** symbol *redirects* the contents of the man page to the **passwd.ps** file. This command is taught in more detail in a following chapter.

3. Using **man**, learn the commands used for viewing and printing PostScript files.
4. Learn how to use the **evince(1)** 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 the **man** command, research **lp(1)** to determine how to print any document starting on a specific page. Without actually entering any commands (because there are no printers), learn the syntax, in one command, to print only pages 2 and 3 of your PostScript file.

7. Using **pinfo**, look for GNU Info documentation about the **evince** viewer.
8. Using Firefox, open the system's package documentation directory and browse into the **man-db** package subdirectory. View the provided manuals.
9. Using the Firefox browser, locate and browse to the **initscripts** package subdirectory. View the **sysconfig.txt** file, which describes important system configuration options stored in the **/etc/sysconfig** directory.

Evaluation

On **workstation**, run **lab help-review grade** to confirm success of this exercise.

```
[student@workstation ~]$ lab help-review grade
```

Finish

On **workstation**, run the **lab help-review finish** script to complete this exercise.

```
[student@workstation ~]$ lab help-review finish
```

This concludes the lab.

► Solution

Getting Help in Red Hat Enterprise Linux

Performance Checklist

In this lab, you will look up information to help you complete tasks in man pages and GNU Info documents.

Outcomes

You should be able to:

- Locate relevant commands by searching man pages and Info nodes.
- Learn new options for commonly used documentation commands.
- Use appropriate tools to view and print documentation and other non-text formatted files.

Before You Begin

Log in to **workstation** as **student** using **student** as the password.

On **workstation**, run the **lab help-review start** command.

```
[student@workstation ~]$ lab help-review start
```

1. On **workstation**, determine how to prepare a man page for printing. Specifically, find what format or rendering language is used for printing.
 - 1.1. Use the **man man** command to determine how to prepare a man page for printing.

```
[student@worksation ~]$ man man  
...output omitted...
```

Press **q** to quit the man page.



Note

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

2. Create a formatted output file of the **passwd** man page. Call the file **passwd.ps**. Determine the file content format. Inspect the contents of the **passwd.ps** file.

**Note**

Create formatted output of the **passwd** man page using the following command:

```
[student@workstation $]$ man -t passwd > passwd.ps
```

The **>** symbol *redirects* the contents of the man page to the **passwd.ps** file. This command is taught in more detail in a following chapter.

- 2.1. Use the **man -t** command to create a formatted file of the **passwd** man page.

```
[student@workstation ~]$ man -t passwd > passwd.ps
[student@workstation ~]$ ls -al
...output omitted...
-rw-rw-r--. 1 student student 19947 Feb 26 11:14 passwd.ps
...output omitted...
```

- 2.2. Use the **file** command to determine the file content format.

```
[student@workstation ~]$ file /home/student/passwd.ps
passwd.ps: PostScript document text conforming DSC level 3.0
```

- 2.3. Use the **less** command to view the **/home/student/passwd.ps** file.

```
[student@workstation ~]$ less /home/student/passwd.ps
%!PS-Adobe-3.0
%%Creator: groff version 1.22.3
%%CreationDate: Tue Feb 26 11:14:40 2019
%%DocumentNeededResources: font Times-Roman
%%+ font Times-Bold
%%+ font Times-Italic
%%+ font Symbol
%%DocumentSuppliedResources: procset grops 1.22 3
...output omitted...
```

**Note**

The output of **file** asserts that the file is in the PostScript format, and you have confirmed it by viewing its contents. Notice the header lines of PostScript information. Use **q** to quit the **less** command.

3. Using **man**, learn the commands used for viewing and printing PostScript files.

- 3.1. Using **man** learn the commands used for viewing and printing PostScript files.

```
[student@workstation ~]# man -k postscript viewer
evince (1) - GNOME document viewer
evince-previewer (1) - show a printing preview of PostScript and PDF documents
evince-thumbnailer (1) - create png thumbnails from PostScript and PDF documents
gcm-viewer (1) - GNOME Color Manager Profile Viewer Tool
```

```
gnome-logs (1)      - log viewer for the systemd journal
grops (1)          - PostScript driver for groff
pango-view (1)      - Pango text viewer
pluginviewer (8)    - list loadable SASL plugins and their properties
```

**Note**

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)** commands in the output.

4. Learn how to use the **evince(1)** viewer in preview mode. Also, determine how to open a document starting on a specific page.

- 4.1. Use the **man evince** command to learn how to use the viewer in preview mode.

```
[student@workstation ~]$ man evince
...output omitted...
```

Press **q** to quit the man page.

**Note**

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

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

- 5.1. Use the **evince** command to open **/home/student/passwd.ps**

```
[student@workstation ~]$ evince /home/student/passwd.ps
```

- 5.2. Use the **evince -w /home/student/passwd.ps** command to open the file in preview mode.

```
[student@workstation ~]$ evince -w /home/student/passwd.ps
```

- 5.3. Use the **evince -i 3 /home/student/passwd.ps** command to open the file at page 3.

```
[student@workstation ~]$ evince -i 3 /home/student/passwd.ps
```

**Note**

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.

6. Using the **man** command, research **lp(1)** to determine how to print any document starting on a specific page. Without actually entering any commands (because there are no printers), learn the syntax, in one command, to print only pages 2 and 3 of your PostScript file.

- 6.1. Use the **man lp** command to determine how to print specific pages of a document.

```
[student@workstation ~]$ man lp  
...output omitted...
```

Press **q** to quit the man page.

**Note**

From **lp(1)**, you 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. Therefore, one valid answer is **lp passwd.ps -P 2-3**.

7. Using **pinfo**, look for GNU Info documentation about the **evince** viewer.

- 7.1. Use the **pinfo command** to look for GNU Info documentation about the **evince** viewer.

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

**Note**

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

8. Using Firefox, open the system's package documentation directory and browse into the **man-db** package subdirectory. View the provided manuals.

- 8.1. Use **firefox /usr/share/doc** to view system documentation. Browse to the **man-db** subdirectory. Click on the manuals to view them.

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

**Note**

Bookmarks can be made for any frequently used directory. 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.

Index of file:///usr/share/doc/man-db/				
Up to higher level directory		Name	Size	Last Modified
		File: ChangeLog	51 KB	12/12/16 1:44:30 PM GMT+1
		File: NEWS	60 KB	11/7/18 4:46:16 PM GMT+1
		File: README	12 KB	12/11/16 12:44:45 AM GMT+1
		man-db-manual.ps	129 KB	11/7/18 4:47:06 PM GMT+1
		man-db-manual.txt	70 KB	11/7/18 4:47:01 PM GMT+1

9. Using the Firefox browser, locate and browse to the **initscripts** package subdirectory. View the **sysconfig.txt** file, which describes important system configuration options stored in the **/etc/sysconfig** directory.
- 9.1. In the Firefox browser, locate the **initscripts** package subdirectory. Notice how useful a browser is for locating and viewing local system documentation. Close the document and Firefox when finished.

Evaluation

On **workstation**, run **lab help-review grade** to confirm success of this exercise.

```
[student@workstation ~]$ lab help-review grade
```

Finish

On **workstation**, run the **lab help-review finish** script to complete this exercise.

```
[student@workstation ~]$ lab help-review finish
```

This concludes the lab.

Summary

In this chapter, you learned:

- Man pages are viewed with the **man** command and provide information on components of a Linux system, such as files, commands, and functions.
- By convention, when referring to a man page the name of a page is followed by its section number in parentheses.
- Info documents are viewed with the **pinfo** command and are made up of a collection of hypertext nodes, providing information about software packages as a whole.
- The navigational keystrokes used by **man** and **pinfo** are slightly different.

Chapter 5

Creating, Viewing, and Editing Text Files

Goal

Create, view, and edit text files from command output or in a text editor.

Objectives

- Save command output or errors to a file with shell redirection, and process command output through multiple command-line programs with pipes.
- Create and edit text files using the **vim** editor.
- Use shell variables to help run commands, and edit Bash startup scripts to set shell and environment variables to modify the behavior of the shell and programs run from the shell.

Sections

- Redirecting Output to a File or Program (and Quiz)
- Editing Text Files from the Shell Prompt (and Guided Exercise)
- Changing the Shell Environment (and Guided Exercise)

Lab

Creating, Viewing, and Editing Text Files

Redirecting Output to a File or Program

Objectives

After completing this section, you should be able to save output or errors to a file with shell redirection, and process command output through multiple command-line programs with pipes.

Standard Input, Standard Output, and Standard Error

A running program, or *process*, needs to read input from somewhere and write output to somewhere. A command run from the shell prompt normally reads its input from the keyboard and sends its output to its terminal window.

A process uses numbered channels called *file descriptors* to get input and send output. All processes start with at least three file descriptors. *Standard input* (channel 0) reads input from the keyboard. *Standard output* (channel 1) sends normal output to the terminal. *Standard error* (channel 2) sends error messages to the terminal. If a program opens separate connections to other files, it may use higher-numbered file descriptors.

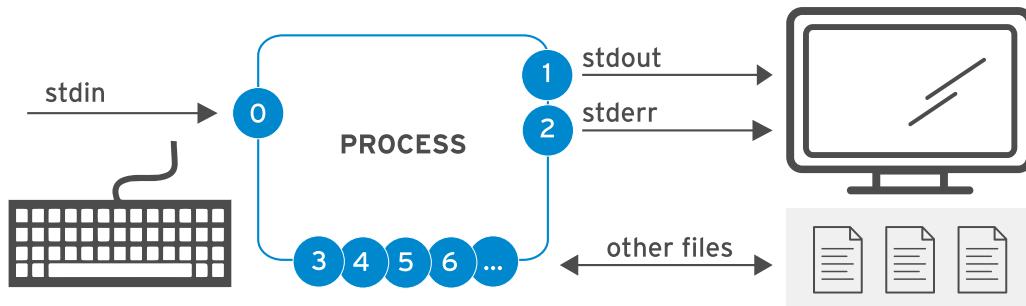


Figure 5.1: Process I/O channels (file descriptors)

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

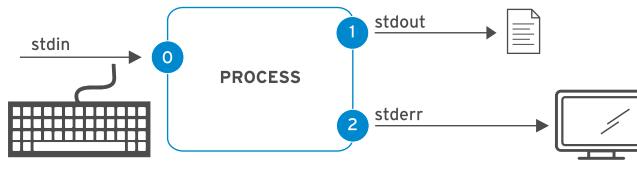
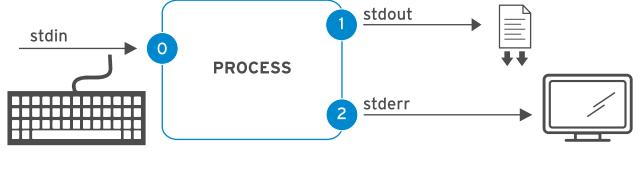
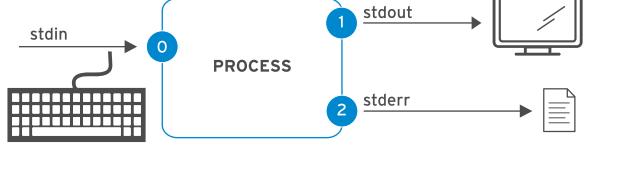
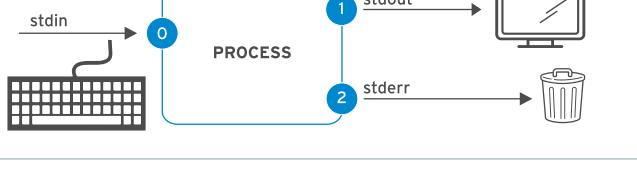
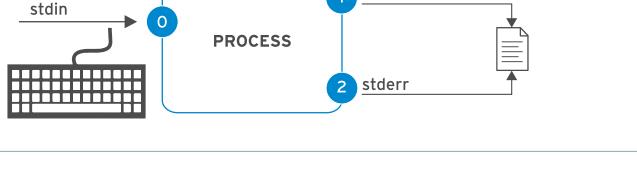
I/O redirection changes how the process gets its input or output. Instead of getting input from the keyboard, or sending output and errors to the terminal, the process reads from or writes to files. Redirection lets you save messages to a file that are normally sent to the terminal window.

Alternatively, you can use redirection to discard output or errors, so they are not displayed on the terminal or saved.

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. If the file does not exist, it will be created. If the file does exist and the redirection is not one that appends to the file, the file's contents will be overwritten.

If you want to discard messages, the special file **/dev/null** quietly discards channel output redirected to it and is always an empty file.

Output Redirection Operators

Usage	Explanation	Visual aid
<code>> file</code>	redirect stdout to overwrite a file	
<code>>> file</code>	redirect stdout to append to a file	
<code>2> file</code>	redirect stderr to overwrite a file	
<code>2> /dev/null</code>	discard stderr error messages by redirecting to /dev/null	
<code>> file 2>&1</code> <code>&> file</code>	redirect stdout and stderr to overwrite the same file	
<code>>> file 2>&1</code> <code>&>> file</code>	redirect stdout and stderr to append to the same file	

Important

The order of redirection operations is important. The following sequence redirects standard output to **file** and then redirects standard error to the same place as standard output (**file**):

```
> file 2>&1
```

However, the next sequence does redirection in the opposite order. This redirects standard error to the default place for standard output (the terminal window, so no change) and *then* redirects only standard output to **file**.

```
2>&1 > file
```

Because of this, some people prefer to use the merging redirection operators:

&>file	instead of	>file 2>&1
----------------------	------------	-----------------------------

&>>file	instead of	>>file 2>&1 (in Bash 4 / RHEL 6 and later)
--------------------------	------------	--

However, other system administrators and programmers who also use other shells related to **bash** (known as Bourne-compatible shells) for scripting commands think that the newer merging redirection operators should be avoided, because they are not standardized or implemented in all of those shells and have other limitations.

The authors of this course take a neutral stance on this topic, and both syntaxes are likely to be encountered in the field.

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 time stamp for later reference.

```
[user@host ~]$ date > /tmp/saved-timestamp
```

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

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

- Concatenate four files into one.

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

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

```
[user@host ~]$ ls -a > /tmp/my-file-names
```

- Append output to an existing file.

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

- The next few commands generate error messages because some system directories are inaccessible to normal users. Observe as the error messages are redirected. Redirect errors to a file while viewing normal command output on the terminal.

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

- Save process output and error messages to separate files.

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

- Ignore and discard error messages.

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

- Store output and generated errors together.

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

- Append output and generated errors to an existing file.

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

Constructing Pipelines

A *pipeline* is a sequence of one or more commands separated by the *pipe* character (|). A pipe connects the standard output of the first command to the standard input of the next command.

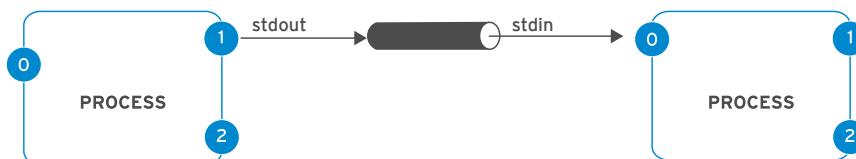


Figure 5.8: Process I/O piping

Pipelines allow the output of a process to be manipulated and formatted by other processes before it is output to the terminal. One useful mental image is to imagine that data is "flowing" through the pipeline from one process to another, being altered slightly by each command in the pipeline through which it flows.



Note

Pipelines and I/O redirection both manipulate standard output and standard input. Redirection sends standard output to files or gets standard input from files. Pipes send the standard output from one process to the standard input of another process.

Pipeline Examples

This example takes the output of the **ls** command and uses **less** to display it on the terminal one screen at a time.

```
[user@host ~]$ ls -l /usr/bin | less
```

The output of the **ls** command is piped to **wc -l**, which counts the number of lines received from **ls** and prints that to the terminal.

```
[user@host ~]$ ls | wc -l
```

In this pipeline, **head** will output the first 10 lines of output from **ls -t**, with the final result redirected to a file.

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

Pipelines, Redirection, and the tee Command

When redirection is combined with a pipeline, the shell sets up the entire pipeline first, then it redirects input/output. If output redirection is used in the *middle* of a pipeline, the output will go to the file and not to the next command in the pipeline.

In this example, the output of the **ls** command goes to the file, and **less** displays nothing on the terminal.

```
[user@host ~]$ ls > /tmp/saved-output | less
```

The **tee** command overcomes this limitation. In a pipeline, **tee** copies its standard input to its standard output and also redirects its standard output to the files named as arguments to the command. If you imagine data as water flowing through a pipeline, **tee** can be visualized as a "T" joint in the pipe which directs output in two directions.

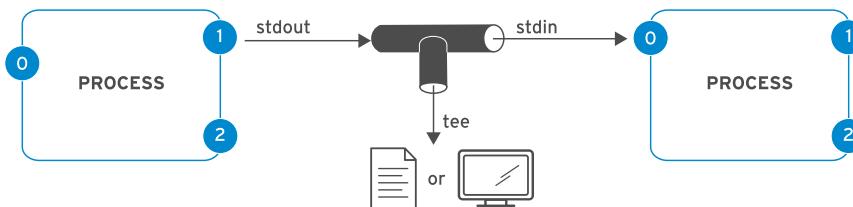


Figure 5.9: Process I/O piping with tee

Pipeline Examples Using the tee Command

This example redirects the output of the **ls** command to the file and passes it to **less** to be displayed on the terminal one screen at a time.

```
[user@host ~]$ ls -l | tee /tmp/saved-output | less
```

If **tee** is used at the end of a pipeline, then the final output of a command can be saved and output to the terminal at the same time.

```
[user@host ~]$ ls -t | head -n 10 | tee /tmp/ten-last-changed-files
```



Important

Standard error can be redirected through a pipe, but the merging redirection operators (`&>` and `&>>`) cannot be used to do this.

The following is the correct way to redirect both standard output and standard error through a pipe:

```
[user@host ~]$ find -name /passwd 2>&1 | less
```



References

info bash (*The GNU Bash Reference Manual*)

- Section 3.2.2: Pipelines
- 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

► Quiz

Redirecting Output to a File or Program

Choose the correct answer to the following questions:

► 1. Which answer displays output to a terminal and ignores all errors?

- a. &>file
- b. 2>>file
- c. 2>/dev/null
- d. 1>/dev/null

► 2. Which answer sends output to a file and sends errors to a different file?

- a. >file 2>file2
- b. >file 1>file2
- c. >file &2>file2
- d. | tee file

► 3. Which answer sends both output and errors to a file, creating it or overwriting its contents?

- a. | tee file
- b. 2 &>file
- c. 1 &>file
- d. &>file

► 4. Which answer sends output and errors to the same file ensuring existing file content is preserved?

- a. >file 2>file2
- b. &>file
- c. >>file 2>&1
- d. >>file 1>&1

► 5. Which answer discards all messages normally sent to the terminal?

- a. >file 2>file2
- b. &>/dev/null
- c. &>/dev/null 2>file
- d. &>file

► **6. Which answer sends output to both the screen and a file at the same time?**

- a. &>/dev/null
- b. >file 2>file2
- c. | tee file
- d. | < file

► **7. Which answer saves output to a file and discards error messages?**

- a. &>file
- b. | tee file 2>/dev/null
- c. > file 1>/dev/null
- d. > file 2>/dev/null

► Solution

Redirecting Output to a File or Program

Choose the correct answer to the following questions:

► 1. Which answer displays output to a terminal and ignores all errors?

- a. &>file
- b. 2>>file
- c. 2>/dev/null
- d. 1>/dev/null

► 2. Which answer sends output to a file and sends errors to a different file?

- a. >file 2>file2
- b. >file 1>file2
- c. >file &2>file2
- d. | tee file

► 3. Which answer sends both output and errors to a file, creating it or overwriting its contents?

- a. | tee file
- b. 2 &>file
- c. 1 &>file
- d. &>file

► 4. Which answer sends output and errors to the same file ensuring existing file content is preserved?

- a. >file 2>file2
- b. &>file
- c. >>file 2>&1
- d. >>file 1>&1

► 5. Which answer discards all messages normally sent to the terminal?

- a. >file 2>file2
- b. &>/dev/null
- c. &>/dev/null 2>file
- d. &>file

► **6. Which answer sends output to both the screen and a file at the same time?**

- a. &>/dev/null
- b. >file 2>file2
- c. | tee file
- d. | < file

► **7. Which answer saves output to a file and discards error messages?**

- a. &>file
- b. | tee file 2>/dev/null
- c. > file 1>/dev/null
- d. > file 2>/dev/null

Editing Text Files from the Shell Prompt

Objectives

After completing this section, you should be able to create and edit text files from the command line using the **vim** editor.

Editing Files with Vim

A key design principle of Linux is that information and configuration settings are commonly stored in text-based files. These files can be structured in various ways, as lists of settings, in INI-like formats, as structured XML or YAML, and so on. However, the advantage of text files is that they 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.

Why Learn Vim?

You should know how to use at least one text editor that can be used from a text-only shell prompt. If you do, you can edit text-based configuration files from a terminal window, or from remote logins through **ssh** or the Web Console. Then you do not need access to a graphical desktop in order to edit files on a server, and in fact that server might not need to run a graphical desktop environment at all.

But then, why learn Vim instead of other possible options? The key reason is that Vim is almost always installed on a server, if any text editor is present. This is because **vi** was specified by the POSIX standard that Linux and many other UNIX-like operating systems comply with in large part.

In addition, Vim is often used as the **vi** implementation on other common operating systems or distributions. For example, macOS currently includes a lightweight installation of Vim by default. So Vim skills learned for Linux might also help you get things done elsewhere.

Starting Vim

Vim may be installed in Red Hat Enterprise Linux in two different ways. This can affect the features and Vim commands available to you.

Your server might only have the *vim-minimal* package installed. This is a very lightweight installation that includes only the core feature set and the basic **vi** command. In this case, you can open a file for editing with **vi filename**, and all the core features discussed in this section will be available to you.

Alternatively, your server might have the *vim-enhanced* package installed. This provides a much more comprehensive set of features, an on-line help system, and a tutorial program. In order to start Vim in this enhanced mode, you use the **vim** command.

```
[user@host ~]$ vim filename
```

Either way, the core features that we will discuss in this section will work with both commands.

**Note**

If `vim-enhanced` is installed, regular users will have a shell alias set so that if they run the `vi` command, they will automatically get the `vim` command instead. This does not apply to `root` and other users with UIDs below 200 (which are used by system services).

If you are editing files as the `root` user and you expect `vi` to run in enhanced mode, this can be a surprise. Likewise, if `vim-enhanced` is installed and a regular user wants the simple `vi` for some reason, they might need to use `\vi` to override the alias temporarily.

Advanced users can use `\vi --version` and `vim --version` to compare the feature sets of the two commands.

Vim Operating Modes

An unusual characteristic of Vim is that it has several *modes* of operation, including *command mode*, *extended command mode*, *edit mode*, and *visual mode*. Depending on the mode, you may be issuing commands, editing text, or working with blocks of text. As a new Vim user, you should always be aware of your current mode as keystrokes have different effects in different modes.

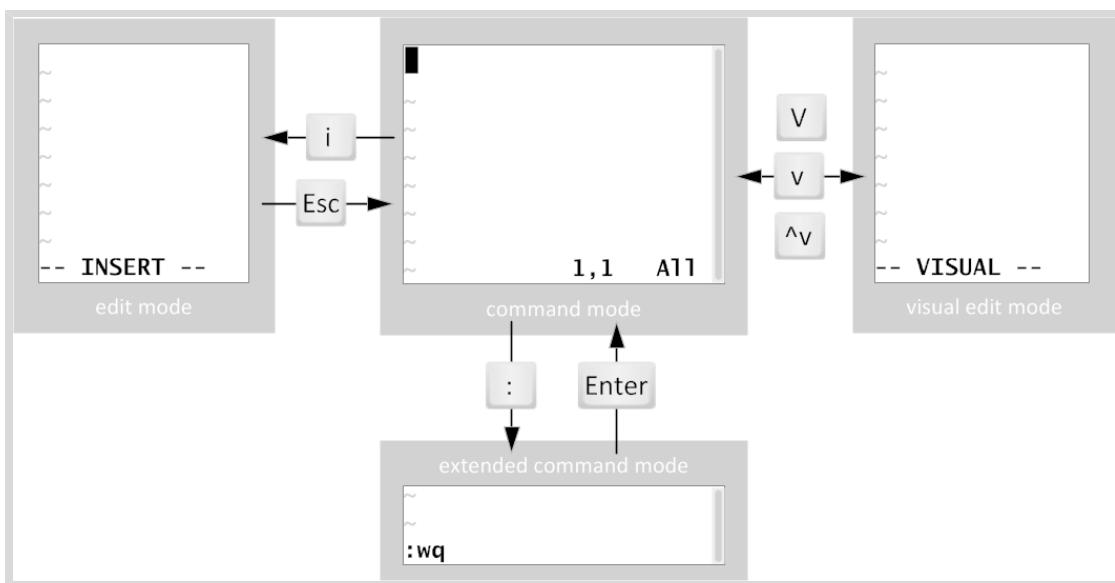


Figure 5.10: Moving between Vim modes

When you first open Vim, it starts in *command mode*, which is 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 `Shift+V` for multiline and `Ctrl+V` for block selection. The same keystroke used to enter *visual mode* (`v`, `Shift+V` or `Ctrl+V`) is used to exit.
- The `:` keystroke begins *extended command mode* for tasks such as writing the file (to save it), and quitting the Vim editor.

**Note**

If you are not sure what mode Vim is in, you can try pressing **Esc** a few times to get back into command mode. Pressing **Esc** in command mode is harmless, so a few extra key presses are okay.

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 **i** key puts Vim into insert mode. All text entered after this is treated as file contents until you exit insert mode. The **Esc** key exits insert mode and returns Vim to command mode. The **u** key will undo the most recent edit. Press the **x** key to delete a single character. The **:w** command writes (saves) the file and remains in command mode for more editing. The **:wq** command writes (saves) the file and quits Vim. The **:q!** command quits Vim, discarding all file changes since the last write. The Vim user must learn these commands to accomplish any editing task.

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, and then enter visual mode. Use the arrow keys to expand the visual selection. When ready, press **y** to *yank* the selection into memory. Position the cursor at the new location, and then press **p** to *put* the selection at the cursor.

Visual Mode in Vim

Visual mode is a great way to highlight and manipulate text. There are three keystrokes:

- Character mode: **v**
- Line mode: **Shift+v**
- Block mode: **Ctrl+v**

Character mode highlights sentences in a block of text. The word **VISUAL** will appear at the bottom of the screen. Press **v** to enter visual character mode. **Shift+v** enters line mode. **VISUAL LINE** will appear at the bottom of the screen.

Visual block mode is perfect for manipulating data files. From the cursor, press the **Ctrl+v** to enter visual block. **VISUAL BLOCK** will appear at the bottom of the screen. Use the arrow keys to highlight the section to change.

**Note**

Vim has a lot of capabilities, but you should master the basic workflow first. You do not need to quickly understand the entire editor and its capabilities. Get comfortable with those basics through practice and then you can expand your Vim vocabulary by learning additional Vim commands (keystrokes).

The exercise for this section will introduce you to the **vimtutor** command. This tutorial, which ships with *vim-enhanced*, is an excellent way to learn the core functionality of Vim.



References

vim(1) man page

The **:help** command in **vim** (if the *vim-enhanced* package is installed).

Vim the editor

<http://www.vim.org/>

Getting Started with Vim visual mode

<https://opensource.com/article/19/2/getting-started-vim-visual-mode>

► Guided Exercise

Editing Text Files from the Shell Prompt

In this exercise, you will use **vimtutor** to practice basic editing techniques in the vim editor.

Outcomes

You should be able to:

- Edit files using Vim.
- Gain competency in Vim using **vimtutor**.

Before You Begin

Log in to **workstation** as **student** using **student** as the password.

On **workstation**, run the **lab edit-vim start** command. This script verifies that the target server is running.

```
[student@workstation ~]$ lab edit-vim start
```

- 1. Use the **ssh** command to log in to **servera**.

```
[student@workstation ~]$ ssh student@servera  
...output omitted...  
[student@servera ~]$
```

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

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

In the presentation, keyboard arrow keys are used for navigation. When **vi** was first developed, users could not rely on having arrow keys or working keyboard mappings for arrow keys to move the cursor. Therefore, **vi** was originally designed to move the cursor using commands using 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.

- 3. In the **vimtutor** window, perform *Lesson 1.2*.

This lesson teaches users how to quit without keeping unwanted changes. All changes are lost. Sometimes this is preferable to leaving a critical file in an incorrect state.

- 4. In the **vimtutor** window, perform *Lesson 1.3*.

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

- 5. In the **vimtutor** window, perform *Lesson 1.4*.

For most editing tasks, the first key pressed is **i**.

- 6. In 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 other available keystrokes to change the cursor placement when insert mode is entered. In insert mode, all typed text is file content.

- 7. In the **vimtutor** window, perform *Lesson 1.6*.

Type **:wq** to save the file and quit the editor.

- 8. In the **vimtutor** window, read the *Lesson 1 Summary*.

The **vimtutor** command includes six more multistep lessons. These lessons are not assigned as part of this course but feel free to explore them on your own to learn more.

- 9. Exit from **servera**.

```
[student@servera ~]$ exit  
logout  
Connection to servera closed.  
[student@workstation ~]$
```

Finish

On **workstation**, run the **lab edit-vim finish** script to complete this exercise.

```
[student@workstation ~]$ lab edit-vim finish
```

This concludes the guided exercise.

Changing the Shell Environment

Objectives

After completing this section, you should be able to set shell variables to help run commands, and edit Bash startup scripts to set shell and environment variables to modify the behavior of the shell and programs run from the shell.

Using Shell Variables

The Bash shell allows you to set *shell variables* that you can use to help run commands or to modify the behavior of the shell. You can also export shell variables as *environment variables*, which are automatically copied to programs run from that shell when they start. You can use variables to help make it easier to run a command with a long argument, or to apply a common setting to commands run from that shell.

Shell variables are unique to a particular shell session. If you have two terminal windows open, or two independent login sessions to the same remote server, you are running two shells. Each shell has its own set of values for its shell variables.

Assigning Values to Variables

Assign a value to a shell variable using the following syntax:

```
VARIABLENAME=value
```

Variable names can contain uppercase or lowercase letters, digits, and the underscore character (_). For example, the following commands set shell variables:

```
[user@host ~]$ COUNT=40
[user@host ~]$ first_name=John
[user@host ~]$ file1=/tmp/abc
[user@host ~]$ _ID=RH123
```

Remember, this change only affects the shell in which you run the command, not any other shells you may be running on that server.

You can use the **set** command to list all shell variables that are currently set. (It also lists all shell functions, which you can ignore.) This list is long enough that you may want to pipe the output into the **less** command so that you can view it one page at a time.

```
[user@host ~]$ set | less
BASH=/usr/bin/bash
BASHOPTS=checkwinsize:cmdhist:complete_fullquote:expand_aliases:extglob:extquote:
force_fignore:histappend:interactive_comments:progcomp:promptvars:sourcepath
BASHRCSCORED=Y
...output omitted...
```

Retrieving Values with Variable Expansion

You can use *variable expansion* to refer to the value of a variable that you have set. To do this, precede the name of the variable with a dollar sign (\$). In the following example, the **echo** command prints out the rest of the command line entered, but after variable expansion is performed.

For example, the following command sets the variable **COUNT** to **40**.

```
[user@host ~]$ COUNT=40
```

If you enter the command **echo COUNT**, it will print out the string **COUNT**.

```
[user@host ~]$ echo COUNT  
COUNT
```

But if you enter the command **echo \$COUNT**, it will print out the value of the variable **COUNT**.

```
[user@host ~]$ echo $COUNT  
40
```

A more practical example might be to use a variable to refer to a long file name for multiple commands.

```
[user@host ~]$ file1=/tmp/tmp.z9pXW0HqcC  
[user@host ~]$ ls -l $file1  
-rw----- 1 student student 1452 Jan 22 14:39 /tmp/tmp.z9pXW0HqcC  
[user@host ~]$ rm $file1  
[user@host ~]$ ls -l $file1  
total 0
```



Important

If there are any trailing characters adjacent to the variable name, you might need to protect the variable name with curly braces. You can always use curly braces in variable expansion, but you will also see many examples in which they are not needed and are omitted.

In the following example, the first **echo** command tries to expand the nonexistent variable **COUNTx**, which does not cause an error but instead returns nothing.

```
[user@host ~]$ echo Repeat $COUNTx  
Repeat  
[user@host ~]$ echo Repeat ${COUNT}x  
Repeat 40x
```

Configuring Bash with Shell Variables

Some shell variables are set when Bash starts but can be modified to adjust the shell's behavior.

For example, two shell variables that affect the shell history and the **history** command are **HISTFILE** and **HISTFILESIZE**. If **HISTFILE** is set, it specifies the location of a file to save

the shell history in when it exits. By default this is the user's `~/.bash_history` file. The **HISTFILESIZE** variable specifies how many commands should be saved in that file from the history.

Another example is **PS1**, which is a shell variable that controls the appearance of the shell prompt. If you change this value, it will change the appearance of your shell prompt. A number of special character expansions supported by the prompt are listed in the "PROMPTING" section of the **bash(1)** man page.

```
[user@host ~]$ PS1="bash\$ "
bash$ PS1="[\u@\\h \\w]\$ "
[user@host ~]$
```

Two items to note about the above example: first, because the value set by PS1 is a prompt, it is virtually always desirable to end the prompt with a trailing space. Second, whenever the value of a variable contains some form of space, including a space, a tab, or a return, the value must be surrounded by quotes, either single or double; this is not optional. Unexpected results will occur if the quotes are omitted. Examine the PS1 example above and note that it conforms to both the recommendation (trailing space) and the rule (quotes).

Configuring Programs with Environment Variables

The shell provides an *environment* to the programs you run from that shell. Among other things, this environment includes information on the current working directory on the file system, the command-line options passed to the program, and the values of *environment variables*. The programs may use these environment variables to change their behavior or their default settings.

Shell variables that are not environment variables can only be used by the shell. Environment variables can be used by the shell *and* by programs run from that shell.



Note

HISTFILE, **HISTFILESIZE**, and **PS1**, learned in the previous section, do not need to be exported as environment variables because they are only used by the shell itself, not by the programs that you run from the shell.

You can make any variable defined in the shell into an environment variable by marking it for export with the **export** command.

```
[user@host ~]$ EDITOR=vim
[user@host ~]$ export EDITOR
```

You can set and export a variable in one step:

```
[user@host ~]$ export EDITOR=vim
```

Applications and sessions use these variables to determine their behavior. For example, the shell automatically sets the **HOME** variable to the file name of the user's home directory when it starts. This can be used to help programs determine where to save files.

Another example is **LANG**, which sets the locale. This adjusts the preferred language for program output; the character set; the formatting of dates, numbers, and currency; and the sort order for programs. If it is set to **en_US.UTF-8**, the locale will use US English with UTF-8 Unicode

character encoding. If it is set to something else, for example **fr_FR.UTF-8**, it will use French UTF-8 Unicode encoding.

```
[user@host ~]$ date  
Tue Jan 22 16:37:45 CST 2019  
[user@host ~]$ export LANG=fr_FR.UTF-8  
[user@host ~]$ date  
mar. janv. 22 16:38:14 CST 2019
```

Another important environment variable is **PATH**. The **PATH** variable contains a list of colon-separated directories that contain programs:

```
[user@host ~]$ echo $PATH  
/home/user/.local/bin:/home/user/bin:/usr/share/Modules/bin:/usr/local/bin:/usr/  
bin:/usr/local/sbin:/usr/sbin
```

When you run a command such as **ls**, the shell looks for the executable file **ls** in each of those directories in order, and runs the first matching file it finds. (On a typical system, this is **/usr/bin/ls**.)

You can easily add additional directories to the end of your **PATH**. For example, perhaps you have executable programs or scripts that you want to run like regular commands in **/home/user/sbin**. You can add **/home/user/sbin** to the end of your **PATH** for the current session like this:

```
[user@host ~]$ export PATH=${PATH}:/home/user/sbin
```

To list all the environment variables for a particular shell, run the **env** command:

```
[user@host ~]$ env  
...output omitted...  
LANG=en_US.UTF-8  
HISTCONTROL=ignoredups  
HOSTNAME=host.example.com  
XDG_SESSION_ID=4  
...output omitted...
```

Setting the Default Text Editor

The **EDITOR** environment variable specifies the program you want to use as your default text editor for command-line programs. Many programs use **vi** or **vim** if it is not specified, but you can override this preference if required:

```
[user@host ~]$ export EDITOR=nano
```



Important

By convention, environment variables and shell variables that are automatically set by the shell have names that use all uppercase characters. If you are setting your own variables, you may want to use names made up of lowercase characters to help avoid naming collisions.

Setting Variables Automatically

If you want to set shell or environment variables automatically when your shell starts, you can edit the Bash startup scripts. When Bash starts, several text files containing shell commands are run which initialize the shell environment.

The exact scripts that run depend on how the shell was started, whether it is an interactive login shell, an interactive non-login shell, or a shell script.

Assuming the default **/etc/profile**, **/etc/bashrc**, and **~/.bash_profile** files, if you want to make a change to your user account that affects all your interactive shell prompts at startup, edit your **~/.bashrc** file. For example, you could set that account's default editor to **nano** by editing the file to read:

```
# .bashrc

# Source global definitions
if [ -f /etc/bashrc ]; then
    . /etc/bashrc
fi

# User specific environment
PATH="$HOME/.local/bin:$HOME/bin:$PATH"
export PATH

# User specific aliases and functions
export EDITOR=nano
```



Note

The best way to adjust settings that affect all user accounts is by adding a file with a name ending in **.sh** containing the changes to the **/etc/profile.d** directory. To do this, you need to be logged in as the **root** user.

Unsetting and Unexporting Variables

To unset and unexport a variable entirely, use the **unset** command:

```
[user@host ~]$ echo $file1
/tmp/tmp.z9pXW0HqcC
[user@host ~]$ unset file1
[user@host ~]$ echo $file1

[user@host ~]$
```

To unexport a variable without unsetting it, use the **export -n** command:

```
[user@host ~]$ export -n PS1
```



References

bash(1), **env(1)**, and **builtins(1)** man pages

► Guided Exercise

Changing the Shell Environment

In this exercise, you will use shell variables and variable expansion to run commands and set an environment variable to adjust the default editor for new shells.

Outcomes:

You should be able to:

- Edit user profile.
- Create a shell variable.
- Create an environment variable.

Before You Begin

Log in to **workstation** as **student** using **student** as the password.

On **workstation**, run the **lab edit-shell start** command. This script verifies that the target server is running.

```
[student@workstation ~]$ lab edit-shell start
```

- 1. Change the student user's **PS1** shell variable to **[\u@\h \t \w]\$** (remember to put the value of **PS1** in quotes and put in a trailing space after the dollar sign). This will add the time to the prompt.

- 1.1. On **workstation**, use the **ssh** command to log in to **servera**.

```
[student@workstation ~]$ ssh student@servera  
...output omitted...  
[student@servera ~]$
```

- 1.2. Use Vim to edit the **~/.bashrc** configuration file.

```
[student@servera ~]$ vim ~/.bashrc
```

- 1.3. Add the **PS1** shell variable and its value to the **~/.bashrc** file. Remember to include a trailing space at the end of the value that you set and put the entire value in quotes, including the trailing space.

```
...output omitted...  
# User specific environment and startup programs  
PATH="$HOME/.local/bin:$HOME/bin:$PATH"  
PS1='[\u@\h \t \w]$ '  
export PATH
```

- 1.4. Exit from **servera** and log in again using the **ssh** command to update the command prompt.

```
[student@servera ~]$ exit  
logout  
Connection to servera closed.  
[student@workstation ~]$ ssh student@servera  
...output omitted...  
[student@servera 14:45:05 ~]$
```

- 2. Assign a value to a local shell variable. Variable names can contain uppercase or lowercase letters, digits, and the underscore character. Retrieve the variable value.

- 2.1. Create a new variable called **file** with a value of **tmp.zdkei083**. The **tmp.zdkei083** file exists in the **student** home directory.

```
[student@servera 14:47:05 ~]$ file=tmp.zdkei083
```

- 2.2. Retrieve the value of the **file** variable.

```
[student@servera 14:48:35 ~]$ echo $file  
tmp.zdkei083
```

- 2.3. Use the variable name **file** and the **ls -l** command to list the **tmp.zdkei083** file. Use the **rm** command and the **file** variable name to delete the **tmp.zdkei083** file. Confirm it has been deleted.

```
[student@servera 14:59:07 ~]$ ls -l $file  
-rw-rw-r--. 1 student student 0 Jan 23 14:59 tmp.zdkei083  
[student@servera 14:59:10 ~]$ rm $file  
[student@servera 14:59:15 ~]$ ls -l $file  
ls: cannot access 'tmp.zdkei083': No such file or directory
```

- 3. Assign a value to the **editor** variable. Use one command to make the variable an environment variable.

```
[student@servera 14:46:40 ~]$ export EDITOR=vim  
[student@servera 14:46:55 ~]$ echo $EDITOR  
vim
```

- 4. Exit from **servera**.

```
[student@servera 14:47:11 ~]$ exit  
logout  
Connection to servera closed.  
[student@workstation ~]$
```

Finish

On **workstation**, run the **lab edit-shell finish** script to complete this exercise.

```
[student@workstation ~]$ lab edit-shell finish
```

This concludes the guided exercise.

▶ Lab

Creating, Viewing, and Editing Text Files

Performance Checklist

In this lab you will edit a text file, using the **vim** editor.

Outcomes

You should be able to:

- Use Vim to perform file editing.
- Use visual mode to simplify file editing.

Before You Begin

Log in to **workstation** as **student** using **student** as the password.

On **workstation**, run the **lab edit-review start** command.

```
[student@workstation ~]$ lab edit-review start
```

1. Redirect a long listing of all content in the student's home directory, including hidden directories and files, into a file named **editing_final_lab.txt**.
2. Edit the file using Vim.
3. Remove the first three lines. Enter line-based visual mode with uppercase **V**.
4. Remove columns on the first line. Enter visual mode with lowercase **v**. Lowercase **v** selects characters on a single line only. The columns after **-rw-** should be deleted.
5. Remove columns, and the subsequent dot (".") on the remaining lines. Use the visual block mode. Enter visual block with the control sequence **Ctrl+V**. Use this key sequence to select a block of characters on multiple lines. The columns after **-rw-** should be deleted.
6. Use visual block mode to remove the fourth column.
7. Use visual block mode to remove the time column, leaving the month and day on all lines.
8. Remove the **Desktop** and **Public** rows. Enter visual line mode with uppercase **V**.
9. Use the **:wq** command to save and exit the file. Make a backup, using the date (in seconds) to create a unique file name.
The following **copy** command is very long and should be entered as a single line.
10. Append a dashed line to the file. The dashed line should contain at least 12 dashes.
The following **echo** command is very long and should be entered on a single line.
11. Append a directory listing of the **Documents** directory. List the directory listing on the terminal and send it to the **editing_final_lab.txt** file with one command line.
12. Confirm that the directory listing is at the bottom of the lab file.

Evaluation

On **workstation**, run the **lab edit-review grade** command to confirm success of this exercise.

```
[student@workstation ~]$ lab edit-review grade
```

Finish

On **workstation**, run the **lab edit-review finish** script to complete this exercise.

```
[student@workstation ~]$ lab edit-review finish
```

This concludes the lab.

► Solution

Creating, Viewing, and Editing Text Files

Performance Checklist

In this lab you will edit a text file, using the **vim** editor.

Outcomes

You should be able to:

- Use Vim to perform file editing.
- Use visual mode to simplify file editing.

Before You Begin

Log in to **workstation** as **student** using **student** as the password.

On **workstation**, run the **lab edit-review start** command.

```
[student@workstation ~]$ lab edit-review start
```

1. Redirect a long listing of all content in the student's home directory, including hidden directories and files, into a file named **editing_final_lab.txt**.



Note

The output may not exactly match the examples shown.

On **workstation**, from the **student** home directory, use the **ls -al** command to redirect a long listing of all content to a file named **editing_final_lab.txt**.

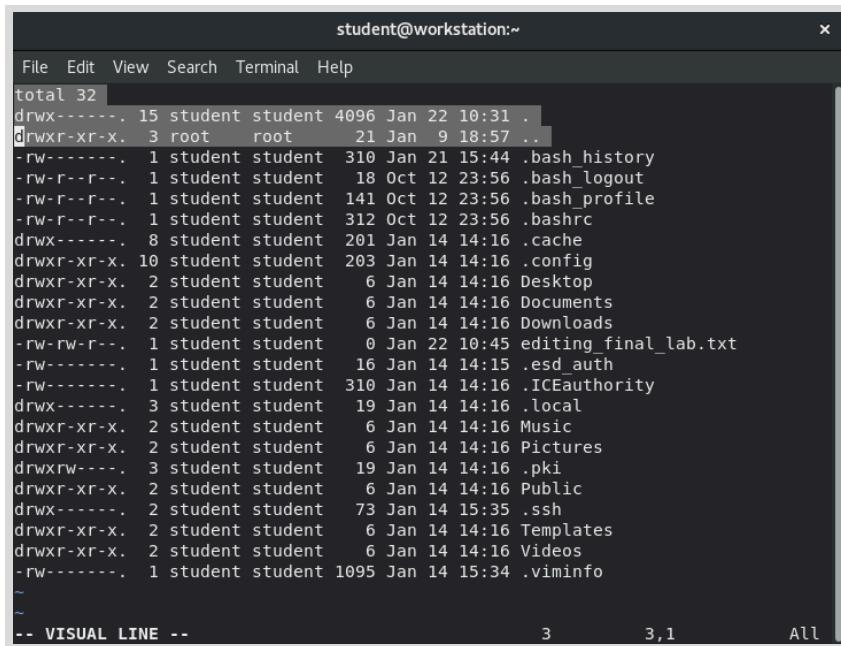
```
[student@workstation ~]$ ls -al > editing_final_lab.txt
```

2. Edit the file using Vim.

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

3. Remove the first three lines. Enter line-based visual mode with uppercase **V**.

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

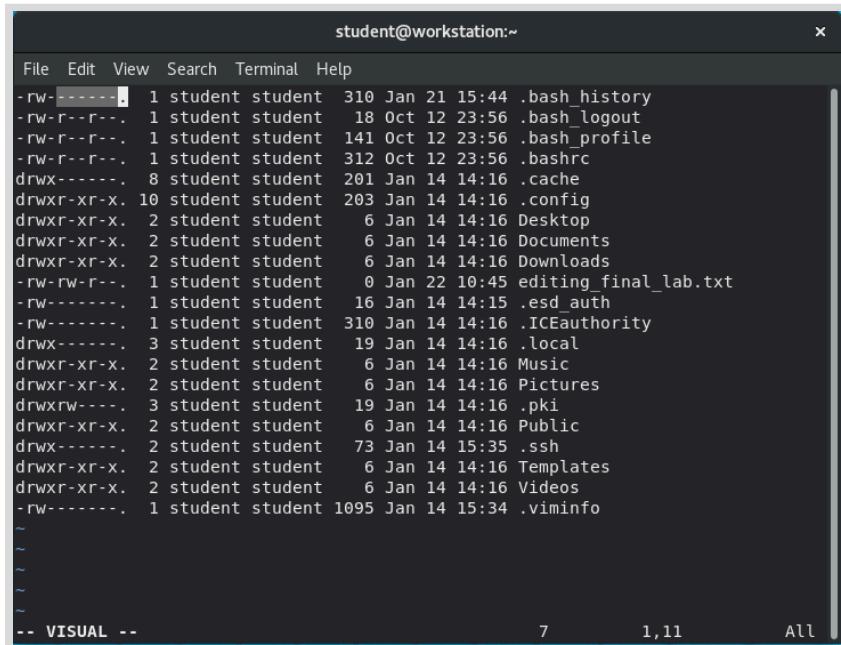


```
student@workstation:~$ ls -l
total 32
drwx----- 15 student student 4096 Jan 22 10:31 .
drwxr-xr-x.  3 root   root    21 Jan  9 18:57 ..
-rw----- 1 student student 310 Jan 21 15:44 .bash_history
-rw-r--r--. 1 student student 18 Oct 12 23:56 .bash_logout
-rw-r--r--. 1 student student 141 Oct 12 23:56 .bash_profile
-rw-r--r--. 1 student student 312 Oct 12 23:56 .bashrc
drwx----- 8 student student 201 Jan 14 14:16 .cache
drwxr-xr-x. 10 student student 203 Jan 14 14:16 .config
drwxr-xr-x.  2 student student  6 Jan 14 14:16 Desktop
drwxr-xr-x.  2 student student  6 Jan 14 14:16 Documents
drwxr-xr-x.  2 student student  6 Jan 14 14:16 Downloads
-rw-rw-r--.  1 student student  0 Jan 22 10:45 editing_final_lab.txt
-rw----- 1 student student 16 Jan 14 14:15 .esd_auth
-rw----- 1 student student 310 Jan 14 14:16 .ICEauthority
drwx----- 3 student student 19 Jan 14 14:16 .local
drwxr-xr-x.  2 student student  6 Jan 14 14:16 Music
drwxr-xr-x.  2 student student  6 Jan 14 14:16 Pictures
drwxrw----. 3 student student 19 Jan 14 14:16 .pki
drwxr-xr-x.  2 student student  6 Jan 14 14:16 Public
drwx----- 2 student student 73 Jan 14 15:35 .ssh
drwxr-xr-x.  2 student student  6 Jan 14 14:16 Templates
drwxr-xr-x.  2 student student  6 Jan 14 14:16 Videos
-rw----- 1 student student 1095 Jan 14 15:34 .viminfo
~
~
-- VISUAL LINE --
```

The terminal window shows a list of files in the current directory. The first line of the output, which is the directory entry for '.', is highlighted with a blue selection bar under the first column of the permissions column. The status bar at the bottom right shows '3' (line number), '3,1' (range), and 'All' (selection mode).

- Remove columns on the first line. Enter visual mode with lowercase **v**. Lowercase **v** selects characters on a single line only. The columns after **-rw-** should be deleted.

Use the arrow keys to position the cursor at the first character. Enter visual mode using lowercase **v**. Use the arrow keys to position the cursor at the last character. Delete the selection with **x**.

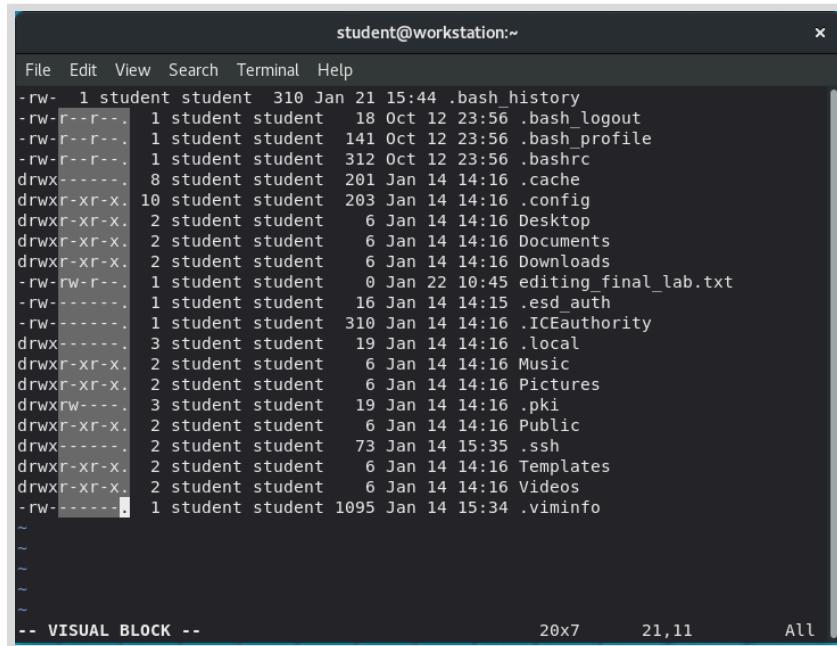


```
student@workstation:~$ ls -l
total 32
-rw----- 1 student student 310 Jan 21 15:44 .bash_history
-rw-r--r--. 1 student student 18 Oct 12 23:56 .bash_logout
-rw-r--r--. 1 student student 141 Oct 12 23:56 .bash_profile
-rw-r--r--. 1 student student 312 Oct 12 23:56 .bashrc
drwx----- 8 student student 201 Jan 14 14:16 .cache
drwxr-xr-x. 10 student student 203 Jan 14 14:16 .config
drwxr-xr-x.  2 student student  6 Jan 14 14:16 Desktop
drwxr-xr-x.  2 student student  6 Jan 14 14:16 Documents
drwxr-xr-x.  2 student student  6 Jan 14 14:16 Downloads
-rw-rw-r--.  1 student student  0 Jan 22 10:45 editing_final_lab.txt
-rw----- 1 student student 16 Jan 14 14:15 .esd_auth
-rw----- 1 student student 310 Jan 14 14:16 .ICEauthority
drwx----- 3 student student 19 Jan 14 14:16 .local
drwxr-xr-x.  2 student student  6 Jan 14 14:16 Music
drwxr-xr-x.  2 student student  6 Jan 14 14:16 Pictures
drwxrw----. 3 student student 19 Jan 14 14:16 .pki
drwxr-xr-x.  2 student student  6 Jan 14 14:16 Public
drwx----- 2 student student 73 Jan 14 15:35 .ssh
drwxr-xr-x.  2 student student  6 Jan 14 14:16 Templates
drwxr-xr-x.  2 student student  6 Jan 14 14:16 Videos
-rw----- 1 student student 1095 Jan 14 15:34 .viminfo
~
~
~
```

The terminal window shows the same file listing as the previous screenshot. The first line of the output is still highlighted with a blue selection bar under the first column of the permissions column. The status bar at the bottom right shows '7' (line number), '1,11' (range), and 'All' (selection mode).

5. Remove columns, and the subsequent dot (".") on the remaining lines. Use the visual block mode. Enter visual block with the control sequence **Ctrl+V**. Use this key sequence to select a block of characters on multiple lines. The columns after -rw- should be deleted.

Use the arrow keys to position the cursor at the first character. Enter visual mode using the control sequence **Ctrl+V**. Use the arrow keys to position the cursor at the last character of the column on the last line. Delete the selection with **X**.

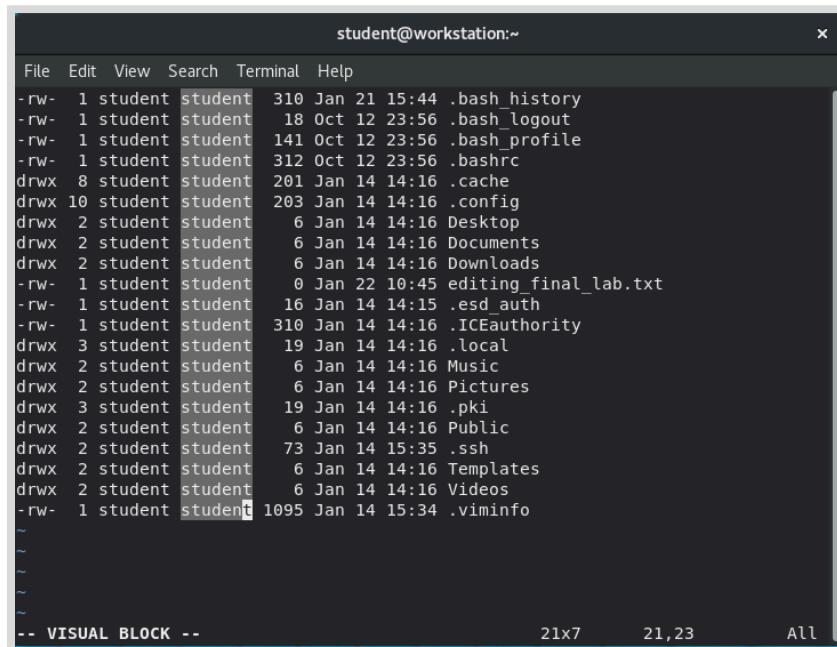


```
student@workstation:~  
File Edit View Search Terminal Help  
-rw- 1 student student 310 Jan 21 15:44 .bash_history  
-rw- 1 student student 18 Oct 12 23:56 .bash_logout  
-rw- 1 student student 141 Oct 12 23:56 .bash_profile  
-rw- 1 student student 312 Oct 12 23:56 .bashrc  
drwx 8 student student 201 Jan 14 14:16 .cache  
drwxr-xr-x. 10 student student 203 Jan 14 14:16 .config  
drwxr-xr-x. 2 student student 6 Jan 14 14:16 Desktop  
drwxr-xr-x. 2 student student 6 Jan 14 14:16 Documents  
drwxr-xr-x. 2 student student 6 Jan 14 14:16 Downloads  
-rw-rw-r--. 1 student student 0 Jan 22 10:45 editing_final_lab.txt  
-rw-----. 1 student student 16 Jan 14 14:15 .esd_auth  
-rw-----. 1 student student 310 Jan 14 14:16 .ICEauthority  
drwx-----. 3 student student 19 Jan 14 14:16 .local  
drwxr-xr-x. 2 student student 6 Jan 14 14:16 Music  
drwxr-xr-x. 2 student student 6 Jan 14 14:16 Pictures  
drwxrwx---. 3 student student 19 Jan 14 14:16 .pki  
drwxr-xr-x. 2 student student 6 Jan 14 14:16 Public  
drwx-----. 2 student student 73 Jan 14 15:35 .ssh  
drwxr-xr-x. 2 student student 6 Jan 14 14:16 Templates  
drwxr-xr-x. 2 student student 6 Jan 14 14:16 Videos  
-rw-----. 1 student student 1095 Jan 14 15:34 .viminfo  
~  
~  
~  
~  
~  
-- VISUAL BLOCK --
```

20x7 21,11 All

6. Use visual block mode to remove the fourth column.

Use the arrow keys to position the cursor at the first character of the fourth column. Enter visual block mode using **Ctrl+V**. Use the arrow keys to position the cursor at the last character and row of the fourth column. Delete the selection with **X**.



```
student@workstation:~  
File Edit View Search Terminal Help  
-rw- 1 student student 310 Jan 21 15:44 .bash_history  
-rw- 1 student student 18 Oct 12 23:56 .bash_logout  
-rw- 1 student student 141 Oct 12 23:56 .bash_profile  
-rw- 1 student student 312 Oct 12 23:56 .bashrc  
drwx 8 student student 201 Jan 14 14:16 .cache  
drwx 10 student student 203 Jan 14 14:16 .config  
drwx 2 student student 6 Jan 14 14:16 Desktop  
drwx 2 student student 6 Jan 14 14:16 Documents  
drwx 2 student student 6 Jan 14 14:16 Downloads  
-rw- 1 student student 0 Jan 22 10:45 editing_final_lab.txt  
-rw- 1 student student 16 Jan 14 14:15 .esd_auth  
-rw- 1 student student 310 Jan 14 14:16 .ICEauthority  
drwx 3 student student 19 Jan 14 14:16 .local  
drwx 2 student student 6 Jan 14 14:16 Music  
drwx 2 student student 6 Jan 14 14:16 Pictures  
drwx 3 student student 19 Jan 14 14:16 .pki  
drwx 2 student student 6 Jan 14 14:16 Public  
drwx 2 student student 73 Jan 14 15:35 .ssh  
drwx 2 student student 6 Jan 14 14:16 Templates  
drwx 2 student student 6 Jan 14 14:16 Videos  
-rw- 1 student student 1095 Jan 14 15:34 .viminfo  
~  
~  
~  
~  
~  
-- VISUAL BLOCK --
```

21x7 21,23 All

7. Use visual block mode to remove the time column, leaving the month and day on all lines.

Use the arrow keys to position the cursor at the first character. Enter visual block mode using **Ctrl+V**. Use the arrow keys to position the cursor at the last character and row of the time column. Delete the selection with **x**.

```
student@workstation:~$ ls -l
-rw- 1 student 310 Jan 21 15:44 .bash_history
-rw- 1 student 18 Oct 12 23:56 .bash_logout
-rw- 1 student 141 Oct 12 23:56 .bash_profile
-rw- 1 student 312 Oct 12 23:56 .bashrc
drwx 8 student 201 Jan 14 14:16 .cache
drwx 10 student 203 Jan 14 14:16 .config
drwx 2 student 6 Jan 14 14:16 Desktop
drwx 2 student 6 Jan 14 14:16 Documents
drwx 2 student 6 Jan 14 14:16 Downloads
-rw- 1 student 0 Jan 22 10:45 editing_final_lab.txt
-rw- 1 student 16 Jan 14 14:15 .esd_auth
-rw- 1 student 310 Jan 14 14:16 .ICEauthority
drwx 3 student 19 Jan 14 14:16 .local
drwx 2 student 6 Jan 14 14:16 Music
drwx 2 student 6 Jan 14 14:16 Pictures
drwx 3 student 19 Jan 14 14:16 .pki
drwx 2 student 6 Jan 14 14:16 Public
drwx 2 student 73 Jan 14 15:35 .ssh
drwx 2 student 6 Jan 14 14:16 Templates
drwx 2 student 6 Jan 14 14:16 Videos
-rw- 1 student 1095 Jan 14 15:34 .viminfo
~
~
~
~
~
-- VISUAL BLOCK --
21x5      21,34      All
```

8. Remove the **Desktop** and **Public** rows. Enter visual line mode with uppercase **V**.

Use the arrow keys to position the cursor at any character on the **Desktop** row. Enter visual mode with uppercase **V**. The full line is selected. Delete the selection with **x**. Repeat for the **Public** row.

```
student@workstation:~$ ls -l
-rw- 1 student 310 Jan 21 .bash_history
-rw- 1 student 18 Oct 12 .bash_logout
-rw- 1 student 141 Oct 12 .bash_profile
-rw- 1 student 312 Oct 12 .bashrc
drwx 8 student 201 Jan 14 .cache
drwx 10 student 203 Jan 14 .config
drwx 2 student 6 Jan 14 Desktop
drwx 2 student 6 Jan 14 Documents
drwx 2 student 6 Jan 14 Downloads
-rw- 1 student 0 Jan 22 editing_final_lab.txt
-rw- 1 student 16 Jan 14 .esd_auth
-rw- 1 student 310 Jan 14 .ICEauthority
drwx 3 student 19 Jan 14 .local
drwx 2 student 6 Jan 14 Music
drwx 2 student 6 Jan 14 Pictures
drwx 3 student 19 Jan 14 .pki
drwx 2 student 6 Jan 14 Public
drwx 2 student 73 Jan 14 .ssh
drwx 2 student 6 Jan 14 Templates
drwx 2 student 6 Jan 14 Videos
-rw- 1 student 1095 Jan 14 .viminfo
~
~
~
~
~
-- VISUAL LINE --
1      7,1      All
```

9. Use the `:wq` command to save and exit the file. Make a backup, using the date (in seconds) to create a unique file name.

```
student@workstation:~
```

File Edit View Search Terminal Help

```
-rw- 1 student 310 Jan 21 .bash_history
-rw- 1 student 18 Oct 12 .bash_logout
-rw- 1 student 141 Oct 12 .bash_profile
-rw- 1 student 312 Oct 12 .bashrc
drwx 8 student 201 Jan 14 .cache
drwx 10 student 203 Jan 14 .config
drwx 2 student 6 Jan 14 Documents
drwx 2 student 6 Jan 14 Downloads
-rw- 1 student 0 Jan 22 editing_final_lab.txt
-rw- 1 student 16 Jan 14 .esd_auth
-rw- 1 student 310 Jan 14 .ICEauthority
drwx 3 student 19 Jan 14 .local
drwx 2 student 6 Jan 14 Music
drwx 2 student 6 Jan 14 Pictures
drwx 3 student 19 Jan 14 .pki
drwx 2 student 73 Jan 14 .ssh
drwx 2 student 6 Jan 14 Templates
drwx 2 student 6 Jan 14 Videos
-rw- 1 student 1095 Jan 14 .viminfo
~
```

:wc

The following **copy** command is very long and should be entered as a single line.

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

- 10.** Append a dashed line to the file. The dashed line should contain at least 12 dashes.

The following **echo** command is very long and should be entered on a single line.

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

11. Append a directory listing of the **Documents** directory. List the directory listing on the terminal and send it to the **editing_final_lab.txt** file with one command line.

```
[student@workstation ~]$ ls Documents/ | tee -a editing_final_lab.txt  
lab review.txt
```

- 12.** Confirm that the directory listing is at the bottom of the lab file.

```
[student@workstation ~]$ cat editing_final_lab.txt
-rw- 1 student 310 Jan 21 .bash_history
-rw- 1 student 18 Oct 12 .bash_logout
-rw- 1 student 141 Oct 12 .bash_profile
-rw- 1 student 312 Oct 12 .bashrc
drwx 8 student 201 Jan 14 .cache
drwx 10 student 203 Jan 14 .config
drwx 2 student 6 Jan 14 Documents
drwx 2 student 6 Jan 14 Downloads
-rw- 1 student 0 Jan 22 editing_final_lab.txt
-rw- 1 student 16 Jan 14 .esd_auth
```

```
-rw- 1 student 310 Jan 14 .ICEauthority
drwx 3 student 19 Jan 14 .local
drwx 2 student 6 Jan 14 Music
drwx 2 student 6 Jan 14 Pictures
drwx 3 student 19 Jan 14 .pki
drwx 2 student 73 Jan 14 .ssh
drwx 2 student 6 Jan 14 Templates
drwx 2 student 6 Jan 14 Videos
-rw- 1 student 1095 Jan 14 .viminfo
-----
lab_review.txt
```

Evaluation

On **workstation**, run the **lab edit-review grade** command to confirm success of this exercise.

```
[student@workstation ~]$ lab edit-review grade
```

Finish

On **workstation**, run the **lab edit-review finish** script to complete this exercise.

```
[student@workstation ~]$ lab edit-review finish
```

This concludes the lab.

Summary

In this chapter, you learned:

- Running programs, or processes, have three standard communication channels, standard input, standard output, and standard error.
- You can use I/O redirection to read standard input from a file or write the output or errors from a process to a file.
- Pipelines can be used to connect standard output from one process to standard input of another process, and can be used to format output or build complex commands.
- You should know how to use at least one command-line text editor, and Vim is generally installed.
- Shell variables can help you run commands and are unique to a particular shell session.
- Environment variables can help you configure the behavior of the shell or the processes it starts.

Chapter 6

Managing Local Users and Groups

Goal

Create, manage, and delete local users and groups and administer local password policies.

Objectives

- Describe the purpose of users and groups on a Linux system.
- Switch to the superuser account to manage a Linux system, and grant other users superuser access using the **sudo** command.
- Create, modify, and delete locally defined user accounts.
- Create, modify, and delete locally defined group accounts.
- Set a password management policy for users, and manually lock and unlock user accounts.

Sections

- Describing Users and Groups Concepts (and Quiz)
- Gaining Superuser Access (and Guided Exercise)
- Managing Local User Accounts (and Guided Exercise)
- Managing Local Group Accounts (and Guided Exercise)
- Managing User Passwords (and Guided Exercise)

Lab

Managing Local Linux Users and Groups

Describing User and Group Concepts

Objectives

After completing this section, you should be able to describe the purpose of users and groups on a Linux system.

What is a User?

A *user account* is used to provide security boundaries between different people and programs that can run commands.

Users have *user names* to identify them to human users and make them easier to work with. Internally, the system distinguishes user accounts by the unique identification number assigned to them, the *user ID* or *UID*. If a user account is used by humans, it will generally be assigned a secret *password* that the user will use to prove that they are the actual authorized user when logging in.

User accounts are fundamental to system security. Every process (running program) on the system runs as a particular user. Every file has a particular user as its owner. File ownership helps the system enforce access control for users of the files. The user associated with a running process determines the files and directories accessible to that process.

There are three main types of user account: the *superuser*, *system users*, and *regular users*.

- The *superuser* account is for administration of the system. The name of the superuser is **root** and the account has UID 0. The superuser has full access to the system.
- The system has *system user* accounts which are used by processes that provide supporting services. These processes, or *daemons*, usually do not need to run as the superuser. They are assigned non-privileged accounts that allow them to secure their files and other resources from each other and from regular users on the system. Users do not interactively log in using a system user account.
- Most users have *regular user* accounts which they use for their day-to-day work. Like system users, regular users have limited access to the system.

You can use the **id** command to show information about the currently logged-in user.

```
[user01@host ~]$ id  
uid=1000(user01) gid=1000(user01) groups=1000(user01)  
context=unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023
```

To view basic information about another user, pass the username to the **id** command as an argument.

```
[user01@host]$ id user02  
uid=1002(user02) gid=1001(user02) groups=1001(user02)  
context=unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023
```

To view the owner of a file use the **ls -l** command. To view the owner of a directory use the **ls -ld** command. In the following output, the third column shows the username.

```
[user01@host ~]$ ls -l file1
-rw-rw-r-- 1 user01 user01 0 Feb  5 11:10 file1
[user01@host]$ ls -ld dir1
drwxrwxr-x. 2 user01 user01 6 Feb  5 11:10 dir1
```

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. In the following output, the first column shows the username.

```
[user01@host]$ ps -au
USER      PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
root      777  0.0  0.0 225752 1496 tty1      Ss+  11:03   0:00 /sbin/agetty -o -
          p -- \u --noclear tty1 linux
root      780  0.0  0.1 225392 2064 ttys0      Ss+  11:03   0:00 /sbin/agetty -o -
          p -- \u --keep-baud 115200,38400,9600
user01    1207  0.0  0.2 234044 5104 pts/0      Ss   11:09   0:00 -bash
user01    1319  0.0  0.2 266904 3876 pts/0      R+   11:33   0:00 ps au
```

The output of the preceding command displays users by name, but internally the operating system uses the UIDs to track users. The mapping of usernames to UIDs is defined in databases of account information. By default, systems use the **/etc/passwd** file to store information about local users.

Each line in the **/etc/passwd** file contains information about one user. It is divided up into seven colon-separated fields. Here is an example of a line from **/etc/passwd**:

```
① user01:②x:③1000:④1000:⑤User One:⑥/home/user01:⑦/bin/bash
```

- ① Username for this user (**user01**).
- ② The user's password used to be stored here in encrypted format. That has been moved to the **/etc/shadow** file, which will be covered later. This field should always be **x**.
- ③ The UID number for this user account (**1000**).
- ④ The GID number for this user account's primary group (**1000**). Groups will be discussed later in this section.
- ⑤ The real name for this user (**User One**).
- ⑥ The home directory for this user (**/home/user01**). This is the initial working directory when the shell starts and contains the user's data and configuration settings.
- ⑦ The default shell program for this user, which runs on login (**/bin/bash**). For a regular user, this is normally the program that provides the user's command-line prompt. A system user might use **/sbin/nologin** if interactive logins are not allowed for that user.

What is a Group?

A group is a collection of users that need to share access to files and other system resources. Groups can be used to grant access to files to a set of users instead of just a single user.

Like users, groups have *group names* to make them easier to work with. Internally, the system distinguishes groups by the unique identification number assigned to them, the *group ID* or *GID*.

The mapping of group names to GIDs is defined in databases of group account information. By default, systems use the **/etc/group** file to store information about local groups.

Each line in the **/etc/group** file contains information about one group. Each group entry is divided into four colon-separated fields. Here is an example of a line from **/etc/group**:

```
❶ group01:❷ x:❸ 10000:❹ user01,user02,user03
```

- ❶ Group name for this group (**group01**).
- ❷ Obsolete group password field. This field should always be **x**.
- ❸ The GID number for this group (**10000**).
- ❹ A list of users who are members of this group as a supplementary group (**user01, user02, user03**). Primary (or default) and supplementary groups are discussed later in this section.

Primary Groups and Supplementary Groups

Every user has exactly one primary group. For local users, this is the group listed by GID number in the **/etc/passwd** file. By default, this is the group that will own new files created by the user.

Normally, when you create a new regular user, a new group with the same name as that user is created. That group is used as the primary group for the new user, and that user is the only member of this *User Private Group*. It turns out that this helps make management of file permissions simpler, which will be discussed later in this course.

Users may also have *supplementary groups*. Membership in supplementary groups is determined by the **/etc/group** file. Users are granted access to files based on whether any of their groups have access. It doesn't matter if the group or groups that have access are primary or supplementary for the user.

For example, if the user **user01** has a primary group **user01** and supplementary groups **wheel** and **webadmin**, then that user can read files readable by any of those three groups.

The **id** command can also be used to find out about group membership for a user.

```
[user03@host ~]$ id
uid=1003(user03) gid=1003(user03) groups=1003(user03),10(wheel),10000(group01)
context=unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023
```

In the preceding example, **user03** has the group **user03** as their primary group (**gid**). The **groups** item lists all groups for this user, and other than the primary group **user03**, the user has groups **wheel** and **group01** as supplementary groups.



References

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

info libc (GNU C Library Reference Manual)

- Section 30: Users and groups

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

► Quiz

Describing User and Group Concepts

Choose the correct answer to the following questions:

- ▶ 1. Which item represents a number that identifies the user at the most fundamental level?
 - a. primary user
 - b. UID
 - c. GID
 - d. username

- ▶ 2. Which item represents the program that provides the user's command-line prompt?
 - a. primary shell
 - b. home directory
 - c. login shell
 - d. command name

- ▶ 3. Which item or file represents the location of the local group information?
 - a. home directory
 - b. **/etc/passwd**
 - c. **/etc/GID**
 - d. **/etc/group**

- ▶ 4. Which item or file represents the location of the user's personal files?
 - a. home directory
 - b. login shell
 - c. **/etc/passwd**
 - d. **/etc/group**

- ▶ 5. Which item represents a number that identifies the group at the most fundamental level?
 - a. primary group
 - b. UID
 - c. GID
 - d. groupid

- ▶ 6. Which item or file represents the location of the local user account information?
 - a. home directory
 - b. **/etc/passwd**
 - c. **/etc/UID**
 - d. **/etc/group**

► **7. What is the fourth field of the /etc/passwd file?**

- a. home directory
- b. UID
- c. login shell
- d. primary group

► Solution

Describing User and Group Concepts

Choose the correct answer to the following questions:

- ▶ **1. Which item represents a number that identifies the user at the most fundamental level?**
 - a. primary user
 - b. UID
 - c. GID
 - d. username

- ▶ **2. Which item represents the program that provides the user's command-line prompt?**
 - a. primary shell
 - b. home directory
 - c. login shell
 - d. command name

- ▶ **3. Which item or file represents the location of the local group information?**
 - a. home directory
 - b. `/etc/passwd`
 - c. `/etc/GID`
 - d. `/etc/group`

- ▶ **4. Which item or file represents the location of the user's personal files?**
 - a. home directory
 - b. login shell
 - c. `/etc/passwd`
 - d. `/etc/group`

- ▶ **5. Which item represents a number that identifies the group at the most fundamental level?**
 - a. primary group
 - b. UID
 - c. GID
 - d. groupid

- ▶ **6. Which item or file represents the location of the local user account information?**
 - a. home directory
 - b. `/etc/passwd`
 - c. `/etc/UID`
 - d. `/etc/group`

► **7. What is the fourth field of the /etc/passwd file?**

- a. home directory
- b. UID
- c. login shell
- d. primary group

Gaining Superuser Access

Objectives

After completing this section, you will be able to switch to the superuser account to manage a Linux system, and grant other users superuser access through the **sudo** command.

The Superuser

Most operating systems have some sort of *superuser*, a user that has all power over the system. In Red Hat Enterprise Linux this 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. To perform tasks such as installing or removing software and to manage system files and directories, users must escalate their privileges to the **root** user.

The **root** user only among normal users can control most devices, but there are a few exceptions. For example, normal users can control removable devices, such as USB devices. Thus, normal users can add and remove files and otherwise manage a removable device, but only **root** can manage "fixed" hard drives by default.

This unlimited privilege, however, comes with responsibility. The **root** user has unlimited power to damage the system: remove files and directories, remove user accounts, add back doors, and so on. If the **root** user's account is compromised, someone else would have administrative control of the system. Throughout this course, administrators are 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 Microsoft Windows. In Linux, most system administrators log in to the system as an unprivileged user and use various tools to temporarily gain **root** privileges.



Warning

One common practice on Microsoft Windows in the past was for the local **Administrator** user to log in directly to perform system administrator duties. Although this is possible on Linux, Red Hat recommends that system administrators do not log in directly as **root**. Instead, system administrators should log in as a normal user and use other mechanisms (**su**, **sudo**, or PolicyKit, for example) to temporarily gain superuser privileges.

By logging in as the superuser, 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.

Switching Users

The **su** command allows users to switch to a different user account. If you run **su** from a regular user account, you will be prompted for the password of the account to which you want to switch. When **root** runs **su**, you do not need to enter the user's password.

```
[user01@host ~]$ su - user02
Password:
[user02@host ~]$
```

If you omit the user name, the **su** or **su -** command attempts to switch to **root** by default.

```
[user01@host ~]$ su -
Password:
[root@host ~]#
```

The command **su** starts a *non-login shell*, while the command **su -** (with the dash option) starts a *login shell*. The main distinction between the two commands is that **su -** sets up the shell environment as if it were a new login as that user, while **su** just starts a shell as that user, but uses the original user's environment settings.

In most cases, administrators should run **su -** to get a shell with the target user's normal environment 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. Run **info su** to view more details.

Running Commands with Sudo

In some cases, the **root** user's account may not have a valid password at all for security reasons. In this case, users cannot log in to the system as **root** directly with a password, and **su** cannot be used to get an interactive shell. One tool that can be used to get **root** access in this case is **sudo**.

Unlike **su**, **sudo** normally requires users to enter their own password for authentication, not the password of the user account they are trying to access. That is, users who use **sudo** to run commands as **root** do not need to know the **root** password. Instead, they use their own passwords to authenticate access.

Additionally, **sudo** can be configured to allow specific users to run any command as some other user, or only some commands as that user.

For example, when **sudo** is configured to allow the **user01** user to run the command **usermod** as **root**, **user01** could run the following command to lock or unlock a user account:

```
[user01@host ~]$ sudo usermod -L user02
[sudo] password for user01:
[user01@host ~]$ su - user02
Password:
su: Authentication failure
[user01@host ~]$
```

If a user tries to run a command as another user, and the **sudo** configuration does not permit it, the command will be blocked, the attempt will be logged, and by default an email will be sent to the **root** user.

```
[user02@host ~]$ sudo tail /var/log/secure
[sudo] password for user02:
user02 is not in the sudoers file. This incident will be reported.
[user02@host ~]$
```

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

```
[user01@host ~]$ sudo tail /var/log/secure
...output omitted...
Feb  6 20:45:46 host sudo[2577]:  user01 : TTY=pts/0 ; PWD=/home/user01 ;
USER=root ; COMMAND=/sbin/usermod -L user02
...output omitted...
```

In Red Hat Enterprise Linux 7 and Red Hat Enterprise Linux 8, all members of the **wheel** group can use **sudo** to run commands as any user, including **root**. The user is prompted for their own password. This is a change from Red Hat Enterprise Linux 6 and earlier, where users who were members of the **wheel** group did not get this administrative access by default.



Warning

RHEL 6 did not grant the **wheel** group any special privileges by default. Sites that have been using this group for a non-standard purpose might be surprised when RHEL 7 and RHEL 8 automatically grants all members of **wheel** full **sudo** privileges. This could lead to unauthorized users getting administrative access to RHEL 7 and RHEL 8 systems.

Historically, UNIX-like systems use membership in the **wheel** group to grant or control superuser access.

Getting an Interactive Root Shell with Sudo

If there is a nonadministrative user account on the system that can use **sudo** to run the **su** command, you can run **sudo su -** from that account to get an interactive **root** user shell. This works because **sudo** will run **su -** as **root**, and **root** does not need to enter a password to use **su**.

Another way to access the **root** account with **sudo** is to use the **sudo -i** command. This will switch to the **root** account and run that user's default shell (usually **bash**) and associated shell login scripts. If you just want to run the shell, you can use the **sudo -s** command.

For example, an administrator might get an interactive shell as **root** on an AWS EC2 instance by using SSH public-key authentication to log in as the normal user **ec2-user**, and then by running **sudo -i** to get the **root** user's shell.

```
[ec2-user@host ~]$ sudo -i
[sudo] password for ec2-user:
[root@host ~]#
```

The **sudo su -** command and **sudo -i** do not behave exactly the same. This will be discussed briefly at the end of the section.

Configuring Sudo

The main configuration file for **sudo** is **/etc/sudoers**. To avoid problems if multiple administrators try to edit it at the same time, it should only be edited with the special **visudo** command.

For example, the following line from the **/etc/sudoers** file enables **sudo** access for members of group **wheel**.

```
%wheel      ALL=(ALL)    ALL
```

In this line, **%wheel** is the user or group to whom the rule applies. A **%** specifies that this is a group, group **wheel**. The **ALL=(ALL)** specifies that on any host that might have this file, **wheel** can run any command. The final **ALL** specifies that **wheel** can run those commands as any user on the system.

By default, **/etc/sudoers** also includes the contents of any files in the **/etc/sudoers.d** directory as part of the configuration file. This allows an administrator to add **sudo** access for a user simply by putting an appropriate file in that directory.



Note

Using supplementary files under the **/etc/sudoers.d** directory is convenient and simple. You can enable or disable **sudo** access simply by copying a file into the directory or removing it from the directory.

In this course, you will create and remove files in the **/etc/sudoers.d** directory to configure **sudo** access for users and groups.

To enable full **sudo** access for the user **user01**, you could create **/etc/sudoers.d/user01** with the following content:

```
user01  ALL=(ALL)  ALL
```

To enable full **sudo** access for the group **group01**, you could create **/etc/sudoers.d/group01** with the following content:

```
%group01  ALL=(ALL)  ALL
```

It is also possible to set up **sudo** to allow a user to run commands as another user without entering their password:

```
ansible  ALL=(ALL)  NOPASSWD:ALL
```

While there are obvious security risks to granting this level of access to a user or group, it is frequently used with cloud instances, virtual machines, and provisioning systems to help configure servers. The account with this access must be carefully protected and might require SSH public-key authentication in order for a user on a remote system to access it at all.

For example, the official AMI for Red Hat Enterprise Linux in the Amazon Web Services Marketplace ships with the **root** and the **ec2-user** users' passwords locked. The **ec2-user** user account is set up to allow remote interactive access through SSH public-key authentication. The

user **ec2-user** can also run any command as **root** without a password because the last line of the AMI's **/etc/sudoers** file is set up as follows:

```
ec2-user  ALL=(ALL)  NOPASSWD:  ALL
```

The requirement to enter a password for **sudo** can be re-enabled or other changes may be made to tighten security as part of the process of configuring the system.



Note

In this course, you may see **sudo su** - used instead of **sudo -i**. Both commands work, but there are some subtle differences between them.

The **sudo su** - command sets up the **root** environment exactly like a normal login because the **su** - command ignores the settings made by **sudo** and sets up the environment from scratch.

The default configuration of the **sudo -i** command actually sets up some details of the **root** user's environment differently than a normal login. For example, it sets the **PATH** environment variable slightly differently. This affects where the shell will look to find commands.

You can make **sudo -i** behave more like **su** - by editing **/etc/sudoers** with **visudo**. Find the line

```
Defaults    secure_path = /sbin:/bin:/usr/sbin:/usr/bin
```

and replace it with the following two lines:

```
Defaults    secure_path = /usr/local/bin:/usr/bin
Defaults>root  secure_path = /usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin
```

For most purposes, this is not a major difference. However, for consistency of **PATH** settings on systems with the default **/etc/sudoers** file, the authors of this course use **sudo -i** in examples and exercises.



References

su(1), **sudo(8)**, **visudo(8)** and **sudoers(5)** man pages

info libc persona (*GNU C Library Reference Manual*)

- Section 30.2: The Persona of a Process

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

► Guided Exercise

Gaining Superuser Access

In this exercise, you will practice switching to the **root** account and running commands as **root**.

Outcomes

You should be able to:

- Use **sudo** to switch to **root** and access the interactive shell as **root** without knowing the password of the superuser.
- Explain how **su** and **sudo** - can affect the shell environment through running or not running the login scripts.
- Use **sudo** to run other commands as **root**.

Before You Begin

Log in to **workstation** as **student** using **student** as the password.

On **workstation**, run **lab users-sudo start** to start the exercise. This script creates the necessary user accounts and files to set up the environment correctly.

```
[student@workstation ~]$ lab users-sudo start
```

- 1. From **workstation**, open an SSH session to **servera** as **student**.

```
[student@workstation ~]$ ssh student@servera  
...output omitted...  
[student@servera ~]$
```

- 2. Explore the shell environment of **student**. View the current user and group information and display the current working directory. Also view the environment variables that specify the user's home directory and the locations of the user's executables.

- 2.1. Run **id** to view the current user and group information.

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

- 2.2. Run **pwd** to display the current working directory.

```
[student@servera ~]$ pwd  
/home/student
```

- 2.3. Print the values of the **HOME** and **PATH** variables to determine the home directory and user executables' path, respectively.

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

- 3. Switch to **root** in a non-login shell and explore the new shell environment.

- 3.1. Run **sudo su** at the shell prompt to become the **root** user.

```
[student@servera ~]$ sudo su  
[sudo] password for student: student  
[root@servera student]#
```

- 3.2. Run **id** to view the current user and group information.

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

- 3.3. Run **pwd** to display the current working directory.

```
[root@servera student]# pwd  
/home/student
```

- 3.4. Print the values of the **HOME** and **PATH** variables to determine the home directory and user executables' path, respectively.

```
[root@servera student]# echo $HOME  
/root  
[root@servera student]# echo $PATH  
/sbin:/bin:/usr/sbin:/usr/bin:/usr/local/sbin:/usr/local/bin
```

If you already have some experience with Linux and the **su** command, you may have expected that using **su** without the dash (-) option to become **root** would cause you to keep the current **PATH** of **student**. That did not happen. As you will see in the next step, this is not the usual **PATH** for **root** either.

What happened? The difference is that you did not run **su** directly. Instead, you ran **su** as **root** using **sudo** because you did not possess the password of the superuser. The **sudo** command initially overrides the **PATH** variable from the initial environment for security reasons. Any command that runs after the initial override can still update the **PATH** variable, as you will see in the following steps.

- 3.5. Exit the **root** user's shell to return to the **student** user's shell.

```
[root@servera student]# exit  
exit  
[student@servera ~]$
```

- 4. Switch to **root** in a login shell and explore the new shell environment.

- 4.1. Run **sudo su -** at the shell prompt to become the **root** user.

```
[student@servera ~]$ sudo su -
[root@servera ~]#
```

Notice the difference in the shell prompt compared to that of **sudo su** in the preceding step.

sudo may or may not prompt you for the **student** password, depending on the time-out period of **sudo**. The default time-out period is five minutes. If you have authenticated to **sudo** within the last five minutes, **sudo** will not prompt you for the password. If it has been more than five minutes since you authenticated to **sudo**, you need to enter **student** as the password to get authenticated to **sudo**.

- 4.2. Run **id** to view the current user and group information.

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

- 4.3. Run **pwd** to display the current working directory.

```
[root@servera ~]# pwd
/root
```

- 4.4. Print the values of the **HOME** and **PATH** variables to determine the home directory and the user executables' path, respectively.

```
[root@servera ~]# echo $HOME
/root
[root@servera ~]# echo $PATH
/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/root/bin
```

As in the preceding step, after **sudo** reset the **PATH** variable from the settings in the **student** user's shell environment, the **su -** command ran the shell login scripts for **root** and set the **PATH** variable to yet another value. The **su** command without the dash (-) option did not do that.

- 4.5. Exit the **root** user's shell to return to the **student** user's shell.

```
[root@servera ~]# exit
logout
[student@servera ~]$
```

- 5. Verify that the **operator1** user is configured as to run any command as any user using **sudo**.

```
[student@servera ~]$ sudo cat /etc/sudoers.d/operator1
operator1 ALL=(ALL) ALL
```

- 6. Become **operator1** and view the contents of **/var/log/messages**. Copy **/etc/motd** to **/etc/motdOLD** and remove it (**/etc/motdOLD**). These operations require administrative rights and so use **sudo** to run those commands as the superuser. Do not switch to root using **sudo su** or **sudo su -**. Use **redhat** as the password of **operator1**.

- 6.1. Switch to **operator1**.

```
[student@servera ~]$ su - operator1  
Password: redhat  
[operator1@servera ~]$
```

- 6.2. Attempt to view the last five lines of **/var/log/messages** without using **sudo**. This should fail.

```
[operator1@servera ~]$ tail -5 /var/log/messages  
tail: cannot open '/var/log/messages' for reading: Permission denied
```

- 6.3. Attempt to view the last five lines of **/var/log/messages** with **sudo**. This should succeed.

```
[operator1@servera ~]$ sudo tail -5 /var/log/messages  
[sudo] password for operator1: redhat  
Jan 23 15:53:36 servera su[2304]: FAILED SU (to operator1) student on pts/1  
Jan 23 15:53:51 servera su[2307]: FAILED SU (to operator1) student on pts/1  
Jan 23 15:53:58 servera su[2310]: FAILED SU (to operator1) student on pts/1  
Jan 23 15:54:12 servera su[2322]: (to operator1) student on pts/1  
Jan 23 15:54:25 servera su[2353]: (to operator1) student on pts/1
```



Note

The preceding output may differ on your system.

- 6.4. Attempt to make a copy of **/etc/motd** as **/etc/motdOLD** without using **sudo**. This should fail.

```
[operator1@servera ~]$ cp /etc/motd /etc/motdOLD  
cp: cannot create regular file '/etc/motdOLD': Permission denied
```

- 6.5. Attempt to make a copy of **/etc/motd** as **/etc/motdOLD** with **sudo**. This should succeed.

```
[operator1@servera ~]$ sudo cp /etc/motd /etc/motdOLD  
[operator1@servera ~]$
```

- 6.6. Attempt to delete **/etc/motdOLD** without using **sudo**. This should fail.

```
[operator1@servera ~]$ rm /etc/motdOLD  
rm: remove write-protected regular empty file '/etc/motdOLD'? y  
rm: cannot remove '/etc/motdOLD': Permission denied  
[operator1@servera ~]$
```

6.7. Attempt to delete **/etc/motdOLD** with **sudo**. This should succeed.

```
[operator1@servera ~]$ sudo rm /etc/motdOLD  
[operator1@servera ~]$
```

6.8. Exit the **operator1** user's shell to return to the **student** user's shell.

```
[operator1@servera ~]$ exit  
logout  
[student@servera ~]$
```

6.9. Log off from **servera**.

```
[student@servera ~]$ exit  
logout  
Connection to servera closed.  
[student@workstation ~]$
```

Finish

On **workstation**, run **lab users-sudo finish** to complete this exercise. This script deletes the user accounts and files created at the start of the exercise to ensure that the environment is clean.

```
[student@workstation ~]$ lab users-sudo finish
```

This concludes the guided exercise.

Managing Local User Accounts

Objectives

After completing this section, you should be able to create, modify, and delete local user accounts.

Managing Local Users

A number of command-line tools can be used to manage local user accounts.

Creating Users from the Command Line

- The **useradd *username*** command creates a new user named **username**. It sets up the user's home directory and account information, and creates a private group for the user named **username**. At this point the account does not have a valid password set, and the user cannot log in until a password is set.
- The **useradd --help** command displays 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 does not affect existing users.

Modifying Existing Users from the Command Line

- The **usermod --help** command displays the basic options that can be used to modify an account. Some common options include:

usermod options:	Usage
-c, --comment COMMENT	Add the user's real name to the comment field.
-g, --gid GROUP	Specify the primary group for the user account.
-G, --groups GROUPS	Specify a comma-separated list of supplementary groups for the user account.
-a, --append	Used with the -G option to add the supplementary groups to the user's current set of group memberships instead of replacing the set of supplementary groups with a new set.
-d, --home HOME_DIR	Specify a particular home directory for the user account.
-m, --move-home	Move the user's home directory to a new location. Must be used with the -d option.
-s, --shell SHELL	Specify a particular login shell for the user account.
-L, --lock	Lock the user account.
-U, --unlock	Unlock the user account.

Deleting Users from the Command Line

- The **userdel username** command removes the details of **username** from **/etc/passwd**, but leaves the user's home directory intact.
- The **userdel -r username** command removes the details of **username** from **/etc/passwd** and also deletes 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 UID. This can also happen when a file, having a deleted user as its owner, exists outside that user's home directory. This situation can lead to information leakage and other security issues.

In Red Hat Enterprise Linux 7 and Red Hat Enterprise Linux 8, the **useradd** command assigns new users the first free UID greater than or equal to 1000, unless you explicitly specify one using the **-u** option.

This is how information leakage can occur. If the first free UID had been previously assigned to a user account which has since been removed from the system, the old user's UID 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@host ~]# useradd user01
[root@host ~]# ls -l /home
drwx----- 3 user01 user01 74 Feb 4 15:22 user01
[root@host ~]# userdel user01
[root@host ~]# ls -l /home
drwx----- 3 1000 1000 74 Feb 4 15:22 user01
[root@host ~]# useradd user02
[root@host ~]# ls -l /home
drwx----- 3 user02 user02 74 Feb 4 15:23 user02
drwx----- 3 user02 user02 74 Feb 4 15:22 user01
```

Notice that **user02** now owns all files that **user01** previously 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 use the **find / -nouser -o -nogroup** command to find all unowned files and directories.

Setting Passwords from the Command Line

- The **passwd username** command sets the initial password or changes the existing password of **username**.
- The **root** user can set a password to any value. A message is 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@host ~]# passwd user01
Changing password for user user01.
New password: redhat
BAD PASSWORD: The password fails the dictionary check - it is based on a
dictionary word
Retype new password: redhat
passwd: all authentication tokens updated successfully.
[root@host ~]#
```

- A regular user must choose a password at least eight characters long and is also 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 only the resources they need to function.
- *UID 1000+* is the range available for assignment to regular users.



Note

Prior to RHEL 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.



References

useradd(8), **usermod(8)**, **userdel(8)** man pages

► Guided Exercise

Managing Local User Accounts

In this exercise, you will create several users on your system and set passwords for those users.

Outcomes

You should be able to configure a Linux system with additional user accounts.

Before You Begin

Log in to **workstation** as **student** using **student** as the password.

On **workstation**, run **lab users-manage start** to start the exercise. This script ensures that the environment is set up correctly.

```
[student@workstation ~]$ lab users-manage start
```

- 1. From **workstation**, open an SSH session to **servera** as **student**.

```
[student@workstation ~]$ ssh student@servera  
...output omitted...  
[student@servera ~]$
```

- 2. On **servera**, switch to **root** using **sudo**, converting to the **root** user's shell environment.

```
[student@servera ~]$ sudo su -  
[sudo] password for student: student  
[root@servera ~]#
```

- 3. Create the **operator1** user and confirm that it exists in the system.

```
[root@servera ~]# useradd operator1  
[root@servera ~]# tail /etc/passwd  
...output omitted...  
operator1:x:1002:1002::/home/operator1:/bin/bash
```

- 4. Set the password for **operator1** to **redhat**.

```
[root@servera ~]# passwd operator1  
Changing password for user operator1.  
New password: redhat  
BAD PASSWORD: The password is shorter than 8 characters  
Retype new password: redhat  
passwd: all authentication tokens updated successfully.
```

- ▶ 5. Create the additional users called **operator2** and **operator3**. Set their passwords to **redhat**.

5.1. Add the **operator2** user. Set the password for **operator2** to **redhat**.

```
[root@servera ~]# useradd operator2
[root@servera ~]# passwd operator2
Changing password for user operator2.
New password: redhat
BAD PASSWORD: The password is shorter than 8 characters
Retype new password: redhat
passwd: all authentication tokens updated successfully.
```

5.2. Add the **operator3** user. Set the password for **operator3** to **redhat**.

```
[root@servera ~]# useradd operator3
[root@servera ~]# passwd operator3
Changing password for user operator3.
New password: redhat
BAD PASSWORD: The password is shorter than 8 characters
Retype new password: redhat
passwd: all authentication tokens updated successfully.
```

- ▶ 6. Update the **operator1** and **operator2** user accounts to include the **Operator One** and **Operator Two** comments, respectively. Verify that the comments are successfully added.

6.1. Run **usermod -c** to update the comments of the **operator1** user account.

```
[root@servera ~]# usermod -c "Operator One" operator1
```

6.2. Run **usermod -c** to update the comments of the **operator2** user account.

```
[root@servera ~]# usermod -c "Operator Two" operator2
```

6.3. Confirm that the comments for each of the **operator1** and **operator2** users are reflected in the user records.

```
[root@servera ~]# tail /etc/passwd
...output omitted...
operator1:x:1002:1002:Operator One:/home/operator1:/bin/bash
operator2:x:1003:1003:Operator Two:/home/operator2:/bin/bash
operator3:x:1004:1004::/home/operator3:/bin/bash
```

- ▶ 7. Delete the **operator3** user along with any personal data of the user. Confirm that the user is successfully deleted.

7.1. Remove the **operator3** user from the system.

```
[root@servera ~]# userdel -r operator3
```

7.2. Confirm that **operator3** is successfully deleted.

```
[root@servera ~]# tail /etc/passwd
...output omitted...
operator1:x:1002:1002:Operator One:/home/operator1:/bin/bash
operator2:x:1003:1003:Operator Two:/home/operator2:/bin/bash
```

Notice that the preceding output does not display the user account information of **operator3**.

- 7.3. Exit the **root** user's shell to return to the **student** user's shell.

```
[root@servera ~]# exit
logout
[student@servera ~]$
```

- 7.4. Log off from **servera**.

```
[student@servera ~]$ exit
logout
Connection to servera closed.
[student@workstation ~]$
```

Finish

On **workstation**, run **lab users-manage finish** to complete this exercise. This script ensures that the environment is clean.

```
[student@workstation ~]$ lab users-manage finish
```

This concludes the guided exercise.

Managing Local Group Accounts

Objectives

After completing this section, students should be able to create, modify, and delete local group accounts.

Managing Local 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.

Creating Groups from the Command Line

- The **groupadd** command creates groups. Without options the **groupadd** command uses the next available GID from the range specified in the **/etc/login.defs** file while creating the groups.
- The **-g** option specifies a particular GID for the group to use.

```
[user01@host ~]$ sudo groupadd -g 10000 group01
[user01@host ~]$ tail /etc/group
...output omitted...
group01:x:10000:
```



Note

Given the automatic creation of user private groups (GID 1000+), it is generally recommended to set aside a range of GIDs to be used for supplementary groups. A higher range will avoid a collision with a system group (GID 0-999).

- The **-r** option creates a system group using a GID from the range of valid system GIDs listed in the **/etc/login.defs** file. The **SYS_GID_MIN** and **SYS_GID_MAX** configuration items in **/etc/login.defs** define the range of system GIDs.

```
[user01@host ~]$ sudo groupadd -r group02
[user01@host ~]$ tail /etc/group
...output omitted...
group01:x:10000:
group02:x:988:
```

Modifying Existing Groups from the Command Line

- The **groupmod** command changes the properties of an existing group. The **-n** option specifies a new name for the group.

```
[user01@host ~]$ sudo groupmod -n group0022 group02
[user01@host ~]$ tail /etc/group
...output omitted...
group0022:x:988:
```

Notice that the group name is updated to **group0022** from **group02**.

- The **-g** option specifies a new GID.

```
[user01@host ~]$ sudo groupmod -g 20000 group0022
[user01@host ~]$ tail /etc/group
...output omitted...
group0022:x:20000:
```

Notice that the GID is updated to **20000** from **988**.

Deleting Groups from the Command Line

- The **groupdel** command removes groups.

```
[user01@host ~]$ sudo groupdel group0022
```



Note

You cannot remove a group if it is the primary group of any existing user. As with **userdel**, check all file systems to ensure that no files remain on the system that are owned by the group.

Changing Group Membership from the Command Line

- The membership of a group is controlled with user management. Use the **usermod -g** command to change a user's primary group.

```
[user01@host ~]$ id user02
uid=1006(user02) gid=1008(user02) groups=1008(user02)
[user01@host ~]$ sudo usermod -g group01 user02
[user01@host ~]$ id user02
uid=1006(user02) gid=10000(group01) groups=10000(group01)
```

- Use the **usermod -aG** command to add a user to a supplementary group.

```
[user01@host ~]$ id user03
uid=1007(user03) gid=1009(user03) groups=1009(user03)
[user01@host ~]$ sudo usermod -aG group01 user03
[user01@host ~]$ id user03
uid=1007(user03) gid=1009(user03) groups=1009(user03),10000(group01)
```



Important

The use of the **-a** option makes **usermod** function in *append* mode. Without **-a**, the user will be removed from any of their current supplementary groups that are not included in the **-G** option's list.



References

group(5), **groupadd(8)**, **groupdel(8)**, and **usermod(8)** man pages

► Guided Exercise

Managing Local Group Accounts

In this exercise, you will create groups, use them as supplementary groups for some users without changing those users' primary groups, and configure one of the groups with sudo access to run commands as **root**.

Outcomes

You should be able to:

- Create groups and use them as supplementary groups.
- Configure sudo access for a group.

Before You Begin

Log in to **workstation** as **student** using **student** as the password.

On **workstation**, run **lab users-group-manage start** to start the exercise. This script creates the necessary user accounts to set up the environment correctly.

```
[student@workstation ~]$ lab users-group-manage start
```

- 1. From **workstation**, open an SSH session to **servera** as **student**.

```
[student@workstation ~]$ ssh student@servera  
...output omitted...  
[student@servera ~]$
```

- 2. On **servera**, switch to **root** using **sudo**, inheriting the full environment of the **root** user.

```
[student@servera ~]$ sudo su -  
[sudo] password for student: student  
[root@servera ~]#
```

- 3. Create the **operators** supplementary group with the GID of 30000.

```
[root@servera ~]# groupadd -g 30000 operators
```

- 4. Create **admin** as an additional supplementary group.

```
[root@servera ~]# groupadd admin
```

- 5. Verify that both the **operators** and **admin** supplementary groups exist.

```
[root@servera ~]# tail /etc/group  
...output omitted...  
operators:x:30000:  
admin:x:30001:
```

- 6. Ensure that the users **operator1**, **operator2** and **operator3** belong to the group **operators**.

- 6.1. Add **operator1**, **operator2**, and **operator3** to **operators**.

```
[root@servera ~]# usermod -aG operators operator1  
[root@servera ~]# usermod -aG operators operator2  
[root@servera ~]# usermod -aG operators operator3
```

- 6.2. Confirm that the users are successfully added to the group.

```
[root@servera ~]# id operator1  
uid=1002(operator1) gid=1002(operator1) groups=1002(operator1),30000(operators)  
[root@servera ~]# id operator2  
uid=1003(operator2) gid=1003(operator2) groups=1003(operator2),30000(operators)  
[root@servera ~]# id operator3  
uid=1004(operator3) gid=1004(operator3) groups=1004(operator3),30000(operators)
```

- 7. Ensure that the users **sysadmin1**, **sysadmin2** and **sysadmin3** belong to the group **admin**. Enable administrative rights for all the group members of **admin**. Verify that any member of **admin** can run administrative commands.

- 7.1. Add **sysadmin1**, **sysadmin2**, and **sysadmin3** to **admin**.

```
[root@servera ~]# usermod -aG admin sysadmin1  
[root@servera ~]# usermod -aG admin sysadmin2  
[root@servera ~]# usermod -aG admin sysadmin3
```

- 7.2. Confirm that the users are successfully added to the group.

```
[root@servera ~]# id sysadmin1  
uid=1005(sysadmin1) gid=1005(sysadmin1) groups=1005(sysadmin1),30001(admin)  
[root@servera ~]# id sysadmin2  
uid=1006(sysadmin2) gid=1006(sysadmin2) groups=1006(sysadmin2),30001(admin)  
[root@servera ~]# id sysadmin3  
uid=1007(sysadmin3) gid=1007(sysadmin3) groups=1007(sysadmin3),30001(admin)
```

- 7.3. Examine **/etc/group** to verify the supplemental group memberships.

```
[root@servera ~]# tail /etc/group  
...output omitted...  
operators:x:30000:operator1,operator2,operator3  
admin:x:30001:sysadmin1,sysadmin2,sysadmin3
```

- 7.4. Create the **/etc/sudoers.d/admin** file such that the members of **admin** have full administrative privileges.

```
[root@servera ~]# echo "%admin ALL=(ALL) ALL" >> /etc/sudoers.d/admin
```

- 7.5. Switch to **sysadmin1** (a member of **admin**) and verify that you can run a **sudo** command as **sysadmin1**.

```
[root@servera ~]# su - sysadmin1
[sysadmin1@servera ~]$ sudo cat /etc/sudoers.d/admin
[sudo] password for sysadmin1: redhat
%admin ALL=(ALL) ALL
```

- 7.6. Exit the **sysadmin1** user's shell to return to the **root** user's shell.

```
[sysadmin1@servera ~]$ exit
logout
[root@servera ~]#
```

- 7.7. Exit the **root** user's shell to return to the **student** user's shell.

```
[root@servera ~]# exit
logout
[student@servera ~]$
```

- 7.8. Log off from **servera**.

```
[student@servera ~]$ exit
logout
Connection to servera closed.
[student@workstation ~]$
```

Finish

On **workstation**, run **lab users-group-manage finish** to complete this exercise. This script deletes the user accounts created at the start of the exercise.

```
[student@workstation ~]$ lab users-group-manage finish
```

This concludes the guided exercise.

Managing User Passwords

Objectives

After completing this section, you should be able to set a password management policy for users, and manually lock and unlock user accounts.

Shadow Passwords and Password Policy

At one time, 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 were moved to a separate `/etc/shadow` file which is readable only by **root**. This new file also allowed password aging and expiration features to be implemented.

Like `/etc/passwd`, each user has a line in the `/etc/shadow` file. A sample line from `/etc/shadow` with its nine colon-separated fields is shown below.

```
① user03:②$6$CSSX...output omitted...:③17933:④0:⑤99999:⑥7:⑦2:⑧18113:⑨
```

- ① Username of the account this password belongs to.
- ② The *encrypted password* of the user. The format of encrypted passwords is discussed later in this section.
- ③ The day on which the password was last changed. This is set in days since 1970-01-01, and is calculated in the UTC time zone.
- ④ The minimum number of days that have to elapse since the last password change before the user can change it again.
- ⑤ The maximum number of days that can pass without a password change before the password expires. An empty field means it does not expire based on time since the last change.
- ⑥ Warning period. The user will be warned about an expiring password when they login for this number of days before the deadline.
- ⑦ Inactivity period. Once the password has expired, it will still be accepted for login for this many days. After this period has elapsed, the account will be locked.
- ⑧ The day on which the account expires. This is set in days since 1970-01-01, and is calculated in the UTC time zone. An empty field means it does not expire on a particular date.
- ⑨ The last field is usually empty and is reserved for future use.

Format of an Encrypted Password

The encrypted password field stores three pieces of information: the *hashing algorithm* used, the *salt*, and the *encrypted hash*. Each piece of information is delimited by the \$ sign.

```
$①6$②CSSXcYG1L/4ZfHr/$③2W6evvJahUfzfHpc9X.45Jc6H30E...output omitted...
```

- ① The hashing algorithm used for this password. The number 6 indicates it is a SHA-512 hash, which is the default in Red Hat Enterprise Linux 8. A 1 would indicate MD5, a 5 SHA-256.
- ② The salt used to encrypt the password. This is originally chosen at random.
- ③ The encrypted hash of the user's password. The salt and the unencrypted password are combined and encrypted to generate the encrypted hash of the password.

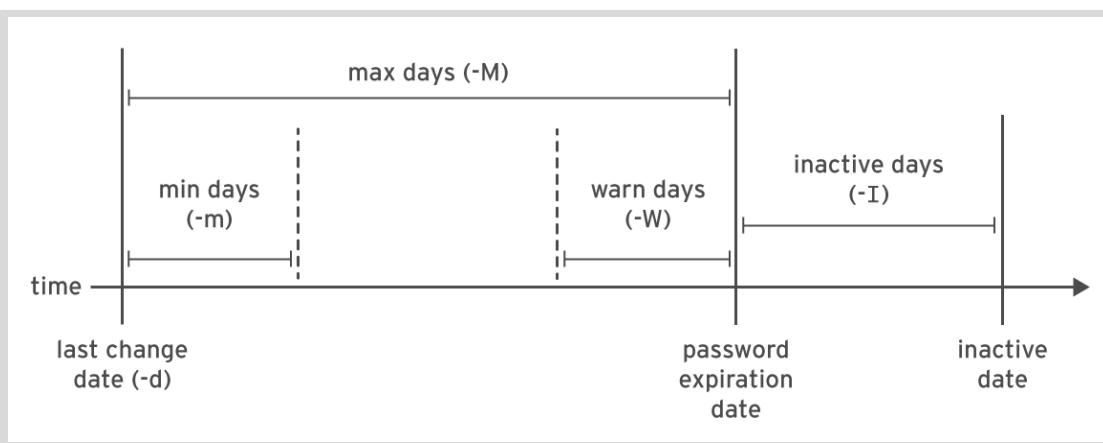
The primary reason to combine a salt with the password is to defend against attacks using pre-computed lists of password hashes. Adding salts changes the resulting hashes, making the pre-computed list useless. If an attacker is able to obtain a copy of an **/etc/shadow** file that is using salts, they will need to perform brute-force password guessing, requiring more time and effort.

Password Verification

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 does not 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.

Configuring Password Aging

The following diagram relates the relevant password aging parameters, which can be adjusted using the **chage** command to implement a password aging policy.



```
[user01@host ~]$ sudo chage -m 0 -M 90 -W 7 -I 14 user03
```

The preceding **chage** command uses the **-m**, **-M**, **-W**, and **-I** options to set the minimum age, maximum age, warning period, and inactivity period of the user's password, respectively.

The **chage -d 0 user03** command forces the **user03** user to update its password on the next login.

The **chage -l user03** command displays the password aging details of **user03**.

The **chage -E 2019-08-05 user03** command causes the **user03** user's account to expire on 2019-08-05 (in YYYY-MM-DD format).



Note

The **date** command can be used to calculate a date in the future. The **-u** option reports the time in UTC.

```
[user01@host ~]$ date -d "+45 days" -u
Thu May 23 17:01:20 UTC 2019
```

Edit the password aging configuration items in the `/etc/login.defs` file to set the default password aging policies. The **PASS_MAX_DAYS** sets the default maximum age of the password. The **PASS_MIN_DAYS** sets the default minimum age of the password. The **PASS_WARN_AGE** sets the default warning period of the password. Any change in the default password aging policies will be effective for new users only. The existing users will continue to use the old password aging settings rather than the new ones.

Restricting Access

You can use the **chage** command to set account expiration dates. When that date is reached, the user cannot log in to the system interactively. The **usermod** command can lock an account with the **-L** option.

```
[user01@host ~]$ sudo usermod -L user03
[user01@host ~]$ su - user03
Password: redhat
su: Authentication failure
```

If a user leaves 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, or in the YYYY-MM-DD format.

```
[user01@host ~]$ sudo usermod -L -e 2019-10-05 user03
```

The preceding **usermod** command uses the **-e** option to set the account expiry date for the given user account. The **-L** option locks the user's password.

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**. If the account was also expired, be sure to also change the expiration date.

The nologin Shell

The **nologin** shell acts as a replacement shell for the user accounts not intended to interactively log into the system. It is wise from a security standpoint to disable an account from logging into the system, when the account does not require it. 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 in to the system directly, the **nologin** shell closes the connection.

```
[user01@host ~]$ usermod -s /sbin/nologin user03
[user01@host ~]$ su - user03
Last login: Wed Feb  6 17:03:06 IST 2019 on pts/0
This account is currently not available.
```



Important

The **nologin** shell prevents interactive use of the system, but does not prevent all access. Users might be able to authenticate and upload or retrieve files through applications such as web applications, file transfer programs, or mail readers if they use the user's password for authentication.



References

chage(1), **usermod(8)**, **shadow(5)**, **crypt(3)** man pages

► Guided Exercise

Managing User Passwords

In this exercise, you will set password policies for several users.

Outcomes

You should be able to:

- Force a password change when the user logs in to the system for the first time.
- Force a password change every 90 days.
- Set the account to expire 180 days from the current day.

Before You Begin

Log in to **workstation** as **student** using **student** as the password.

On **workstation**, run **lab users-pw-manage start** to start the exercise. This script creates the necessary user accounts and files to ensure that the environment is set up correctly.

```
[student@workstation ~]$ lab users-pw-manage start
```

- 1. From **workstation**, open an SSH session to **servera** as **student**.

```
[student@workstation ~]$ ssh student@servera  
...output omitted...  
[student@servera ~]$
```

- 2. On **servera**, explore locking and unlocking user accounts as **student**.

- 2.1. As **student**, lock the **operator1** account using administrative rights.

```
[student@servera ~]$ sudo usermod -L operator1  
[sudo] password for student:
```

- 2.2. Attempt to log in as **operator1**. This should fail.

```
[student@servera ~]$ su - operator1  
Password: redhat  
su: Authentication failure
```

- 2.3. Unlock the **operator1** account.

```
[student@servera ~]$ sudo usermod -U operator1
```

- 2.4. Attempt to log in as **operator1** again. This should succeed.

```
[student@servera ~]$ su - operator1  
Password: redhat  
...output omitted...  
[operator1@servera ~]$
```

- 2.5. Exit out of the **operator1** user's shell to return to the **student** user's shell.

```
[operator1@servera ~]$ exit  
logout
```

- ▶ 3. Change the password policy for **operator1** to require a new password every 90 days. Confirm that the password age is successfully set.

- 3.1. Set the maximum age of the **operator1** user's password to 90 days.

```
[student@servera ~]$ sudo chage -M 90 operator1
```

- 3.2. Verify that the **operator1** user's password expires 90 days after it is changed.

```
[student@servera ~]$ sudo chage -l operator1  
Last password change      : Jan 25, 2019  
Password expires          : Apr 25, 2019  
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
```

- ▶ 4. Force a password change on the first login for the **operator1** account.

```
[student@servera ~]$ sudo chage -d 0 operator1
```

- ▶ 5. Log in as **operator1** and change the password to **forsooth123**. After setting the password, return to the **student** user's shell.

- 5.1. Log in as **operator1** and change the password to **forsooth123** when prompted.

```
[student@servera ~]$ su - operator1  
Password: redhat  
You are required to change your password immediately (administrator enforced)  
Current password: redhat  
New password: forsooth123  
Retype new password: forsooth123  
...output omitted...  
[operator1@servera ~]$
```

- 5.2. Exit the **operator1** user's shell to return to the **student** user's shell.

```
[operator1@servera ~]$ exit
logout
```

- 6. Set the **operator1** account to expire 180 days from the current day. Hint: The **date -d "+180 days"** gives you the date and time 180 days from the current date and time.
- 6.1. Determine a date 180 days in the future. Use the format %F with the **date** command to get the exact value.

```
[student@servera ~]$ date -d "+180 days" +%F
2019-07-24
```

You may get a different value to use in the following step based on the current date and time in your system.

- 6.2. Set the account to expire on the date displayed in the preceding step.

```
[student@servera ~]$ sudo chage -E 2019-07-24 operator1
```

- 6.3. Verify that the account expiry date is successfully set.

```
[student@servera ~]$ sudo chage -l operator1
Last password change      : Jan 25, 2019
Password expires          : Apr 25, 2019
Password inactive         : never
Account expires          : Jul 24, 2019
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
```

- 7. Set the passwords to expire 180 days from the current date for all users. Use administrative rights to edit the configuration file.

- 7.1. Set **PASS_MAX_DAYS** to **180** in **/etc/login.defs**. Use administrative rights when opening the file with the text editor. You can use the **sudo vim /etc/login.defs** command to perform this step.

```
...output omitted...
# Password aging controls:
#
#      PASS_MAX_DAYS    Maximum number of days a password may be
#      used.
#      PASS_MIN_DAYS    Minimum number of days allowed between
#      password changes.
#      PASS_MIN_LEN     Minimum acceptable password length.
#      PASS_WARN_AGE    Number of days warning given before a
#      password expires.
#
PASS_MAX_DAYS    180
PASS_MIN_DAYS    0
```

```
PASS_MIN_LEN      5  
PASS_WARN_AGE     7  
...output omitted...
```



Important

The default password and account expiry settings will be effective for new users but not for existing users.

7.2. Log off from **servera**.

```
[student@servera ~]$ exit  
logout  
Connection to servera closed.  
[student@workstation ~]$
```

Finish

On **workstation**, run **lab users-pw-manage finish** to complete this exercise. This script deletes the user accounts and files created at the start of the exercise to ensure that the environment is clean.

```
[student@workstation ~]$ lab users-pw-manage finish
```

This concludes the guided exercise.

▶ Lab

Managing Local Users and Groups

Performance Checklist

In this lab you will set a default local password policy, create a supplementary group for three users, allow that group to use **sudo** to run commands as **root**, and modify the password policy for one user.

Outcomes

You should be able to:

- Set a default password aging policy of the local user's password.
- Create a group and use the group as a supplementary group for new users.
- Create three new users with the new group as their supplementary group.
- Configure the group members of the supplementary group to run any command as any user using **sudo**.
- Set a user-specific password aging policy.

Before You Begin

Log in to **workstation** as **student** using **student** as the password.

On **workstation**, run **lab users-review start** to start the exercise. This script creates the necessary files to ensure that the environment is set up correctly.

```
[student@workstation ~]$ lab users-review start
```

1. From **workstation**, open an SSH session to **serverb** as **student**.
2. On **serverb**, ensure that newly created users have passwords that must be changed every 30 days.
3. Create the new group called **consultants** with a GID of **35000**.
4. Configure administrative rights for all members of **consultants** to be able to execute any command as any user.
5. Create the **consultant1**, **consultant2**, and **consultant3** users with **consultants** as their supplementary group.
6. Set the **consultant1**, **consultant2**, and **consultant3** accounts to expire in 90 days from the current day.
7. Change the password policy for the **consultant2** account to require a new password every 15 days.
8. Additionally, force the **consultant1**, **consultant2**, and **consultant3** users to change their passwords on the first login.

Evaluation

On **workstation**, run the **lab users-review grade** command to confirm success of this exercise.

```
[student@workstation ~]$ lab users-review grade
```

Finish

On **workstation**, run **lab users-review finish** to complete this lab. This script deletes the user accounts and files created throughout the lab to ensure that the environment is clean.

```
[student@workstation ~]$ lab users-review finish
```

This concludes the lab.

► Solution

Managing Local Users and Groups

Performance Checklist

In this lab you will set a default local password policy, create a supplementary group for three users, allow that group to use **sudo** to run commands as **root**, and modify the password policy for one user.

Outcomes

You should be able to:

- Set a default password aging policy of the local user's password.
- Create a group and use the group as a supplementary group for new users.
- Create three new users with the new group as their supplementary group.
- Configure the group members of the supplementary group to run any command as any user using **sudo**.
- Set a user-specific password aging policy.

Before You Begin

Log in to **workstation** as **student** using **student** as the password.

On **workstation**, run **lab users-review start** to start the exercise. This script creates the necessary files to ensure that the environment is set up correctly.

```
[student@workstation ~]$ lab users-review start
```

1. From **workstation**, open an SSH session to **serverb** as **student**.

```
[student@workstation ~]$ ssh student@serverb
...output omitted...
[student@serverb ~]$
```

2. On **serverb**, ensure that newly created users have passwords that must be changed every 30 days.
 - 2.1. Set **PASS_MAX_DAYS** to **30** in **/etc/login.defs**. Use administrative rights while opening the file with the text editor. You can use the **sudo vim /etc/login.defs** command to perform this step. Use **student** as the password when **sudo** prompts you to enter the **student** user's password.

```
...output omitted...
# Password aging controls:
#
```

```
#      PASS_MAX_DAYS   Maximum number of days a password may be
#      used.
#      PASS_MIN_DAYS   Minimum number of days allowed between
#      password changes.
#      PASS_MIN_LEN     Minimum acceptable password length.
#      PASS_WARN_AGE    Number of days warning given before a
#      password expires.
#
PASS_MAX_DAYS  30
PASS_MIN_DAYS  0
PASS_MIN_LEN   5
PASS_WARN_AGE  7
...output omitted...
```

3. Create the new group called **consultants** with a GID of **35000**.

```
[student@serverb ~]$ sudo groupadd -g 35000 consultants
```

4. Configure administrative rights for all members of **consultants** to be able to execute any command as any user.
 - 4.1. Create the new file **/etc/sudoers.d/consultants** and add the following content to it. You can use the **sudo vim /etc/sudoers.d/consultants** command to perform this step.

```
%consultants  ALL=(ALL) ALL
```

5. Create the **consultant1**, **consultant2**, and **consultant3** users with **consultants** as their supplementary group.

```
[student@serverb ~]$ sudo useradd -G consultants consultant1
[student@serverb ~]$ sudo useradd -G consultants consultant2
[student@serverb ~]$ sudo useradd -G consultants consultant3
```

6. Set the **consultant1**, **consultant2**, and **consultant3** accounts to expire in 90 days from the current day.
 - 6.1. Determine the date 90 days in the future. You may get a different value as compared to the following output based on the current date and time of your system.

```
[student@serverb ~]$ date -d "+90 days" +%F
2019-04-28
```

- 6.2. Set the account expiry date of the **consultant1**, **consultant2**, and **consultant3** accounts to the same value as determined in the preceding step.

```
[student@serverb ~]$ sudo chage -E 2019-04-28 consultant1
[student@serverb ~]$ sudo chage -E 2019-04-28 consultant2
[student@serverb ~]$ sudo chage -E 2019-04-28 consultant3
```

7. Change the password policy for the **consultant2** account to require a new password every 15 days.

```
[student@serverb ~]$ sudo chage -M 15 consultant2
```

8. Additionally, force the **consultant1**, **consultant2**, and **consultant3** users to change their passwords on the first login.
 - 8.1. Set the last day of the password change to **0** so that the users are forced to change the password whenever they log in to the system for the first time.

```
[student@serverb ~]$ sudo chage -d 0 consultant1  
[student@serverb ~]$ sudo chage -d 0 consultant2  
[student@serverb ~]$ sudo chage -d 0 consultant3
```

- 8.2. Log off from **serverb**.

```
[student@serverb ~]$ exit  
logout  
Connection to serverb closed.
```

Evaluation

On **workstation**, run the **lab users-review grade** command to confirm success of this exercise.

```
[student@workstation ~]$ lab users-review grade
```

Finish

On **workstation**, run **lab users-review finish** to complete this lab. This script deletes the user accounts and files created throughout the lab to ensure that the environment is clean.

```
[student@workstation ~]$ lab users-review finish
```

This concludes the lab.

Summary

In this chapter, you learned:

- There are three main types of user account: the superuser, system users, and regular users.
- A user must have a primary group and may be a member of one or more supplementary groups.
- The three critical files containing user and group information are **/etc/passwd**, **/etc/group**, and **/etc/shadow**.
- The **su** and **sudo** commands can be used to run commands as the superuser.
- The **useradd**, **usermod**, and **userdel** commands can be used to manage users.
- The **groupadd**, **groupmod**, and **groupdel** commands can be used to manage groups.
- The **chage** command can be used to configure and view password expiration settings for users.

Chapter 7

Controlling Access to Files

Goal

Set Linux file-system permissions on files and to interpret the security effects of different permission settings.

Objectives

- List the file system permissions on files and directories, and interpret the effect of those permissions on access by users and groups.
- Change the permissions and ownership of files using command-line tools.
- Control the default permissions of new files created by users, explain the effect of special permissions, and use special permissions and default permissions to set the group owner of files created in a particular directory.

Sections

- Interpreting Linux File System Permissions (and Quiz)
- Managing File System Permissions from the Command Line (and Guided Exercise)
- Managing Default Permissions and File Access (and Guided Exercise)

Lab

Controlling Access to Files

Interpreting Linux File System Permissions

Objectives

After completing this section, you should be able to list file-system permissions on files and directories, and interpret the effect of those permissions on access by users and groups.

Linux File-system Permissions

File permissions control access to files. Linux file permissions are simple but flexible, easy to understand and apply, yet still able to handle most normal permission cases easily.

Files have three user categories 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 take precedence. *User* permissions override *group* permissions, which override *other* permissions.

In *Figure 7.1*, **joshua** is a member of the groups **joshua** and **web**, and **allison** is a member of **allison**, **wheel**, and **web**. When **joshua** and **allison** need to collaborate, the files should be associated with the group **web** and group permissions should allow the desired access.

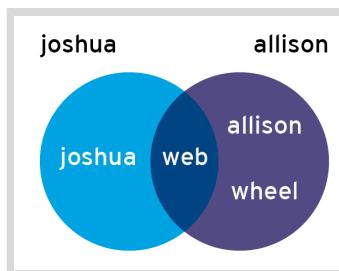


Figure 7.1: Example group membership to facilitate collaboration

Three permission categories apply: read, write, and execute. The following table explains how these permissions affect access to files and directories.

Effects of Permissions on Files and Directories

Permission	Effect on files	Effect on directories
r (read)	File contents can be read.	Contents of the directory (the file names) can be listed.
w (write)	File contents can be changed.	Any file in the directory can be created or deleted.

Permission	Effect on files	Effect on directories
x (execute)	Files can be executed as commands.	The directory can become the current working directory. (You can cd into it, but also require read permission to list files found there.)

Users normally have both read and execute permissions on read-only directories so that they can list the directory and have full read-only access to 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 execute access on a directory, they cannot list file names in the directory. If they know the name of a file that they have permission to read, they can access the contents of that file from outside the directory by explicitly specifying the relative file name.

A file may be removed by anyone who has ownership of, or 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*, discussed later in this chapter.



Note

Linux file permissions work differently than the permissions system used by the NTFS file system for Microsoft Windows.

On Linux, permissions apply only to the file or directory on which they are set. That is, permissions on a directory are not inherited automatically by the subdirectories and files within it. However, permissions on a directory can block access to the contents of the directory depending on how restrictive they are.

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 file or subdirectory owner may delete it, which is similar to the Windows **Write** permission behavior.

The Linux root user has the equivalent of the Windows **Full Control** permission on all files. However, root may have access restricted by the system's SELinux policy using process and file security contexts. SELinux will be discussed in a later course.

Viewing File and Directory Permissions and Ownership

The **-l** option of the **ls** command shows detailed information about permissions and ownership:

```
[user@host ~]$ ls -l test
-rw-rw-r-- 1 student student 0 Feb  8 17:36 test
```

Use the **-d** option to show detailed information about a directory itself, and not its contents.

```
[user@host ~]$ ls -ld /home
drwxr-xr-x. 5 root root 4096 Jan 31 22:00 /home
```

The first character of the long listing is the file type, interpreted like this:

- - is a regular file.
- **d** is a directory.
- **l** is a soft link.
- Other characters represent hardware devices (**b** and **c**) or other special-purpose files (**p** and **s**).

The next nine characters are the file permissions. These are in three sets of three characters: permissions that apply to the user that owns the file, the group that owns the file, and all other users. If the set shows **rwx**, that category has all three permissions, read, write, and execute. If a letter has been replaced by -, then that category does not have that permission.

After the link count, the first name specifies the user that owns the file, and the second name the group that owns the file.

So in the example above, the permissions for user **student** are specified by the first set of three characters. User **student** has read and write on **test**, but not execute.

Group **student** is specified by the second set of three characters: it also has read and write on **test**, but not execute.

Any other user's permissions are specified by the third set of three characters: they only have read permission on **test**.

The most specific set of permissions apply. So if user **student** has different permissions than group **student**, and user **student** is also a member of that group, then the user permissions will be the ones that apply.

Examples of Permission Effects

The following examples will help illustrate how file permissions interact. For these examples, we have four users with the following group memberships:

User	Group Memberships
operator1	operator1, consultant1
database1	database1, consultant1
database2	database2, operator2
contractor1	contractor1, operator2

Those users will be working with files in the **dir** directory. This is a long listing of the files in that directory:

```
[database1@host dir]$ ls -la
total 24
drwxrwxr-x.  2 database1 consultant1  4096 Apr  4 10:23 .
drwxr-xr-x. 10 root      root        4096 Apr  1 17:34 ..
-rw-rw-r--.  1 operator1 operator1    1024 Apr  4 11:02 lfile1
-rw-r--rw-.  1 operator1 consultant1  3144 Apr  4 11:02 lfile2
-rw-rw-r--.  1 database1 consultant1 10234 Apr  4 10:14 rfile1
-rw-r-----. 1 database1 consultant1   2048 Apr  4 10:18 rfile2
```

The **-a** option shows the permissions of hidden files, including the special files used to represent the directory and its parent. In this example, . reflects the permissions of **dir** itself, and .. the permissions of its parent directory.

What are the permissions of **rfile1**? The user that owns the file (**database1**) has read and write but not execute. The group that owns the file (**consultant1**) has read and write but not execute. All other users have read but not write or execute.

The following table explores some of the effects of this set of permissions for these users:

Effect	Why is this true?
The user operator1 can change the contents of rfile1 .	User operator1 is a member of the consultant1 group, and that group has both read and write permissions on rfile1 .
The user database1 can view and modify the contents of rfile2 .	User database1 owns the file and has both read and write access to rfile2 .
The user operator1 can view but not modify the contents of rfile2 (without deleting it and recreating it).	User operator1 is a member of the consultant1 group, and that group only has read access to rfile2 .
The users database2 and contractor1 do not have any access to the contents of rfile2 .	other permissions apply to users database2 and contractor1 , and those permissions do not include read or write permission.
operator1 is the only user who can change the contents of lfile1 (without deleting it and recreating it).	User and group operator1 have write permission on the file, other users do not. But the only member of group operator1 is user operator1 .
The user database2 can change the contents of lfile2 .	User database2 is not the user that owns the file and is not in group consultant1 , so other permissions apply. Those grant write permission.
The user database1 can view the contents of lfile2 , but cannot modify the contents of lfile2 (without deleting it and recreating it).	User database1 is a member of the group consultant1 , and that group only has read permissions on lfile2 . Even though other has write permission, the group permissions take precedence.
The user database1 can delete lfile1 and lfile2 .	User database1 has write permissions on the directory containing both files (shown by .), and therefore can delete any file in that directory. This is true even if database1 does not have write permission on the file itself.



References

[ls\(1\) man page](#)

[info coreutils \(GNU Coreutils\)](#)

- Section 13: Changing file attributes

► Quiz

Interpreting Linux File System Permissions

Review the following information and use it to answer the quiz questions.

The system has four users assigned to the following groups:

- User **consultant1** is in groups **consultant1** and **database1**
- User **operator1** is in groups **operator1** and **database1**
- User **contractor1** is in groups **contractor1** and **contractor3**
- User **operator2** is in groups **operator2** and **contractor3**

The current directory (.) contains four files with the following permissions information:

```
drwxrwxr-x. operator1 database1 .
-rw-rw-r--. consultant1 consultant1 lfile1
-rw-r--rw-. consultant1 database1 lfile2
-rw-rw-r--. operator1 database1 rfile1
-rw-r-----. operator1 database1 rfile2
```

► 1. Which regular file is owned by **operator1** and readable by all users?

- a. **lfile1**
- b. **lfile2**
- c. **rfile1**
- d. **rfile2**

► 2. Which file can be modified by the **contractor1** user?

- a. **lfile1**
- b. **lfile2**
- c. **rfile1**
- d. **rfile2**

► 3. Which file cannot be read by the **operator2** user?

- a. **lfile1**
- b. **lfile2**
- c. **rfile1**
- d. **rfile2**

► **4. Which file has a group ownership of consultant1?**

- a. **lfile1**
- b. **lfile2**
- c. **rfile1**
- d. **rfile2**

► **5. Which files can be deleted by the operator1 user?**

- a. **rfile1**
- b. **rfile2**
- c. All of the above.
- d. None of the above.

► **6. Which files can be deleted by the operator2 user?**

- a. **lfile1**
- b. **lfile2**
- c. All of the above.
- d. None of the above.

► Solution

Interpreting Linux File System Permissions

Review the following information and use it to answer the quiz questions.

The system has four users assigned to the following groups:

- User **consultant1** is in groups **consultant1** and **database1**
- User **operator1** is in groups **operator1** and **database1**
- User **contractor1** is in groups **contractor1** and **contractor3**
- User **operator2** is in groups **operator2** and **contractor3**

The current directory (.) contains four files with the following permissions information:

```
drwxrwxr-x. operator1 database1 .
-rw-rw-r--. consultant1 consultant1 lfile1
-rw-r--rw-. consultant1 database1 lfile2
-rw-rw-r--. operator1 database1 rfile1
-rw-r-----. operator1 database1 rfile2
```

► 1. Which regular file is owned by operator1 and readable by all users?

- a. **lfile1**
- b. **lfile2**
- c. **rfile1**
- d. **rfile2**

► 2. Which file can be modified by the contractor1 user?

- a. **lfile1**
- b. **lfile2**
- c. **rfile1**
- d. **rfile2**

► 3. Which file cannot be read by the operator2 user?

- a. **lfile1**
- b. **lfile2**
- c. **rfile1**
- d. **rfile2**

► **4. Which file has a group ownership of consultant1?**

- a. **lfile1**
- b. **lfile2**
- c. **rfile1**
- d. **rfile2**

► **5. Which files can be deleted by the operator1 user?**

- a. **rfile1**
- b. **rfile2**
- c. All of the above.
- d. None of the above.

► **6. Which files can be deleted by the operator2 user?**

- a. **lfile1**
- b. **lfile2**
- c. All of the above.
- d. None of the above.

Managing File System Permissions from the Command Line

Objectives

After completing this section, you should be able to change the permissions and ownership of files using command-line tools.

Changing File and Directory Permissions

The command used to change permissions from the command line is **chmod**, which means "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).

Changing Permissions with the Symbolic Method

```
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, execute*)

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, you can change one or more of the existing permissions. Use **+** or **-** to add or remove permissions, respectively, or use **=** 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. When using **chmod** to change permissions with the symbolic method, using a capital **X** as the permission flag will add execute permission only if the file is a directory or already has execute set for user, group, or other.

**Note**

The **chmod** command supports the **-R** option to recursively set permissions on the files in an entire directory tree. When using the **-R** option, it can be useful to set permissions symbolically using the **X** option. This allows the execute (search) permission to be set on directories so that their contents can be accessed, without changing permissions on most files. Be cautious with the **X** option, however, because if a file has any execute permission set, **X** will set the specified execute permission on that file as well. For example, the following command recursively sets read and write access on **demodir** and all its children for their group owner, but only applies group execute permissions to directories and files that already have execute set for user, group, or other.

```
[root@host opt]# chmod -R g+rwx demodir
```

Examples

- Remove read and write permission for group and other on **file1**:

```
[user@host ~]$ chmod go-rw file1
```

- Add execute permission for everyone on **file2**:

```
[user@host ~]$ chmod a+x file2
```

Changing Permissions with the Numeric Method

In the example below the # character represents a digit.

```
chmod ### file|directory
```

- Each digit represents permissions for an access level: user, group, other.
- The digit is calculated by adding together numbers for each permission you want to add, 4 for read, 2 for write, and 1 for execute.

Using the *numeric* method, permissions are represented by a 3-digit (or 4-digit, when setting advanced permissions) *octal* number. A single octal digit can represent any single value from 0-7.

In the 3-digit octal (numeric) representation of permissions, each digit stands for one access level, from left to right: user, group, and other. To determine each digit:

- Start with 0.
- If the read permission should be present for this access level, add 4.
- If the write permission should be present, add 2.
- If the execute permission should be present, add 1.

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----**.

Experienced administrators often use numeric permissions because they are shorter to type and pronounce, while still giving full control over all permissions.

Examples

- Set read and write permissions for user, read permission for group and other, on **samplefile**:

```
[user@host ~]$ chmod 644 samplefile
```

- Set read, write, and execute permissions for user, read and execute permissions for group, and no permission for other on **sampledir**:

```
[user@host ~]$ chmod 750 sampledir
```

Changing File and Directory User or Group Ownership

A newly created file is owned by the user who creates that file. By default, new files have a group ownership that is the primary group of the user creating the file. In Red Hat Enterprise Linux, a user's primary group is usually a private group with only that user as a member. To grant access to a file based on group membership, the group that owns the file may need to be changed.

Only **root** can change the user that owns a file. Group ownership, however, can be set by **root** or by the file's owner. **root** can grant file ownership to any group, but regular users can make a group the owner of a file only if they are a member of that group.

File ownership can be changed with the **chown** (change owner) command. For example, to grant ownership of the **test_file** file to the **student** user, use the following command:

```
[root@host ~]# chown student test_file
```

chown can be used with the **-R** option to recursively change the ownership of an entire directory tree. The following command grants ownership of **test_dir** and all files and subdirectories within it to **student**:

```
[root@host ~]# chown -R student test_dir
```

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 changes the group ownership of the **test_dir** directory to **admins**:

```
[root@host ~]# chown :admins test_dir
```

The **chown** command can also be used to change both owner and group at the same time by using the **owner:group** syntax. For example, to change the ownership of **test_dir** to **visitor** and the group to **guests**, use the following command:

```
[root@host ~]# chown visitor:guests test_dir
```

Instead of using **chown**, some users change the group ownership by using the **chgrp** command. This command works just like **chown**, except that it is only used to change group ownership and the colon (:) before the group name is not required.



Important

You may encounter examples of **chown** commands using an alternative syntax that separates owner and group with a period instead of a colon:

```
[root@host ~]# chown owner.group filename
```

You should not use this syntax. Always use a colon.

A period is a valid character in a user name, but a colon is not. If the user **enoch.root**, the user **enoch**, and the group **root** exist on the system, the result of **chown enoch.root filename** will be to have **filename** owned by the user **enoch.root**. You may have been trying to set the file ownership to the user **enoch** and group **root**. This can be confusing.

If you always use the **chown** colon syntax when setting the user and group at the same time, the results are always easy to predict.



References

ls(1), **chmod(1)**, **chown(1)**, and **chgrp(1)** man pages

► Guided Exercise

Managing File System Permissions from the Command Line

In this exercise, you will use file system permissions to create a directory in which all members of a particular group can add and delete files.

Outcomes

You should be able to create a collaborative directory that is accessible by all members of a particular group.

Before You Begin

Log in to **workstation** as **student** using **student** as the password.

On **workstation**, run the **lab perms-cli start** command. The start script creates a group called **consultants** and two users called **consultant1** and **consultant2**.

```
[student@workstation ~]$ lab perms-cli start
```

- 1. From **workstation**, use the **ssh** command to log in to **servera** as the **student** user.

```
[student@workstation ~]$ ssh student@servera  
...output omitted...  
[student@servera ~]$
```

- 2. Switch to the **root** user using **redhat** as the password.

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

- 3. Use the **mkdir** command to create the **/home/consultants** directory.

```
[root@servera ~]# mkdir /home/consultants
```

- 4. Use the **chown** command to change the group ownership of the **consultants** directory to **consultants**.

```
[root@servera ~]# chown :consultants /home/consultants
```

- 5. Ensure that the permissions of the **consultants** group allow its group members to create files in, and delete files from the **/home/consultants** directory. The permissions should forbid others from accessing the files.

- 5.1. Use the **ls** command to confirm that the permissions of the **consultants** group allow its group members to create files in, and delete files from the **/home/consultants** directory.

```
[root@servera ~]# ls -ld /home/consultants
drwxr-xr-x. 2 root     consultants      6 Feb  1 12:08 /home/consultants
```

Note that the **consultants** group currently does not have write permission.

- 5.2. Use the **chmod** command to add write permission to the **consultants** group.

```
[root@servera ~]# chmod g+w /home/consultants
[root@servera ~]# ls -ld /home/consultants
drwxrwxr-x. 2 root consultants 6 Feb  1 13:21 /home/consultants
```

- 5.3. Use the **chmod** command to forbid others from accessing files in the **/home/consultants** directory.

```
[root@servera ~]# chmod 770 /home/consultants
[root@servera ~]# ls -ld /home/consultants
drwxrwx---. 2 root consultants 6 Feb  1 12:08 /home/consultants/
```

- 6. Exit the root shell and switch to the **consultant1** user. The password is **redhat**.

```
[root@servera ~]# exit
logout
[student@servera ~]$
[student@servera ~]$ su - consultant1
Password: redhat
```

- 7. Navigate to the **/home/consultants** directory and create a file called **consultant1.txt**.

- 7.1. Use the **cd** command to change to the **/home/consultants** directory.

```
[consultant1@servera ~]$ cd /home/consultants
```

- 7.2. Use the **touch** command to create an empty file called **consultant1.txt**.

```
[consultant1@servera consultants]$ touch consultant1.txt
```

- 8. Use the **ls -l** command to list the default user and group ownership of the new file and its permissions.

```
[consultant1@servera consultants]$ ls -l consultant1.txt
-rw-rw-r--. 1 consultant1 consultant1 0 Feb  1 12:53 consultant1.txt
```

- 9. Ensure that all members of the **consultants** group can edit the **consultant1.txt** file. Change the group ownership of the **consultant1.txt** file to **consultants**.

- 9.1. Use the **chown** command to change the group ownership of the **consultant1.txt** file to **consultants**.

```
[consultant1@servera consultants]$ chown :consultants consultant1.txt
```

- 9.2. Use the **ls** command with the **-l** option to list the new ownership of the **consultant1.txt** file.

```
[consultant1@servera consultants]$ ls -l consultant1.txt
-rw-rw-r--. 1 consultant1 consultants 0 Feb 1 12:53 consultant1.txt
```

- 10. Exit the shell and switch to the **consultant2** user. The password is **redhat**.

```
[consultant1@servera consultants]$ exit
logout
[student@servera ~]$ su - consultant2
Password: redhat
[consultant2@servera ~]$
```

- 11. Navigate to the **/home/consultants** directory. Ensure that the **consultant2** user can add content to the **consultant1.txt** file. Exit from the shell.

- 11.1. Use the **cd** command to change to the **/home/consultants** directory. Use the **echo** command to add **text** to the **consultant1.txt** file.

```
[consultant2@servera ~]$ cd /home/consultants/
[consultant2@servera consultants]$ echo "text" >> consultant1.txt
[consultant2@servera consultants]$
```

- 11.2. Use the **cat** command to verify that the text was added to the **consultant1.txt** file.

```
[consultant2@servera consultants]$ cat consultant1.txt
text
[consultant2@servera consultants]$
```

- 11.3. Exit the shell.

```
[consultant2@servera consultants]$ exit
logout
[student@servera ~]$
```

- 12. Log off from **servera**.

```
[student@servera ~]$ exit
logout
Connection to servera closed.
[student@workstation ~]$
```

Finish

On **workstation**, run the **lab perms-cli finish** script to complete this exercise.

```
[student@workstation ~]$ lab perms-cli finish
```

This concludes the guided exercise.

Managing Default Permissions and File Access

Objectives

After completing this section, students should be able to:

- Control the default permissions of new files created by users.
- Explain the effect of special permissions.
- Use special permissions and default permissions to set the group owner of files created in a particular directory.

Special Permissions

Special permissions constitute a fourth permission type in addition to the basic user, group, and other types. As the name implies, these permissions provide additional access-related features over and above what the basic permission types allow. This section details the impact of special permissions, summarized in the table below.

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 access to the directory can only remove files that they own; they cannot remove or force saves to files owned by other users.

The *setuid* permission on an executable file means that commands run as the user owning the file, not as the user that ran the command. One example is the **passwd** command:

```
[user@host ~]$ ls -l /usr/bin/passwd
-rwsr-xr-x. 1 root root 35504 Jul 16 2010 /usr/bin/passwd
```

In a long listing, you can identify the setuid permissions by a lowercase **s** where you would normally expect the **x** (owner execute permissions) to be. If the owner does not have execute permissions, this is replaced by an uppercase **S**.

The special permission *setgid* on a directory means that files created in the directory inherit their group ownership 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

Chapter 7 | Controlling Access to Files

private group to the shared group, or if files in a directory should be always owned by a specific group. An example of this is the **/run/log/journal** directory:

```
[user@host ~]$ ls -ld /run/log/journal  
drwxr-sr-x. 3 root systemd-journal 60 May 18 09:15 /run/log/journal
```

If setgid is set on an executable file, commands run as the group that owns that file, not as the user that ran the command, in a similar way to setuid works. One example is the **locate** command:

```
[user@host ~]$ ls -ld /usr/bin/locate  
-rwx--s--x. 1 root slocate 47128 Aug 12 17:17 /usr/bin/locate
```

In a long listing, you can identify the setgid permissions by a lowercase **s** where you would normally expect the **x** (group execute permissions) to be. If the group does not have execute permissions, this is replaced by an uppercase **S**.

Lastly, 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**:

```
[user@host ~]$ ls -ld /tmp  
drwxrwxrwt. 39 root root 4096 Feb 8 20:52 /tmp
```

In a long listing, you can identify the sticky permissions by a lowercase **t** where you would normally expect the **x** (other execute permissions) to be. If other does not have execute permissions, this is replaced by an uppercase **T**.

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**:

```
[user@host ~]# chmod g+s directory
```

- Set the setgid bit and add read/write/execute permissions for user and group, with no access for others, on **directory**:

```
[user@host ~]# chmod 2770 directory
```

Default File Permissions

When you create a new file or directory, it is assigned initial permissions. There are two things that affect these initial permissions. The first is whether you are creating a regular file or a directory. The second is the current *umask*.

If you create a new directory, the operating system starts by assigning it octal permissions 0777 (**drwxrwxrwx**). If you create a new regular file, the operating system assigns it octal permissions 0666 (-**r-w-rw-**). You always have to explicitly add execute permission to a regular file. This

makes it harder for an attacker to compromise a network service so that it creates a new file and immediately executes it as a program.

However, the shell session will also set a umask to further restrict the permissions that are initially set. This is an octal bitmask used to clear the permissions of new files and directories created by a process. If a bit is set in the umask, then the corresponding permission is cleared on new files. For example, the 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 0077 clears all the group and other permissions of newly created files.

The **umask** command without arguments will display the current value of the shell's umask:

```
[user@host ~]$ umask  
0002
```

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. You can omit any leading zeros in the umask.

The system's default umask values for Bash shell users are defined in the **/etc/profile** and **/etc/bashrc** files. Users can override the system defaults in the **.bash_profile** and **.bashrc** files in their home directories.

umask Example

The following example explains how the umask affects the permissions of files and directories. Look at the default umask permissions for both files and directories in the current shell. The owner and group both have read and write permission on files, and other is set to read. The owner and group both have read, write, and execute permissions on directories. The only permission for other is read.

```
[user@host ~]$ umask  
0002  
[user@host ~]$ touch default  
[user@host ~]$ ls -l default.txt  
-rw-rw-r--. 1 user user 0 May  9 01:54 default.txt  
[user@host ~]$ mkdir default  
[user@host ~]$ ls -ld default  
drwxrwxr-x. 2 user user 0 May  9 01:54 default
```

By setting the umask value to 0, the file permissions for other change from read to read and write. The directory permissions for other changes from read and execute to read, write, and execute.

```
[user@host ~]$ umask 0  
[user@host ~]$ touch zero.txt  
[user@host ~]$ ls -l zero.txt  
-rw-rw-rw-. 1 user user 0 May  9 01:54 zero.txt  
[user@host ~]$ mkdir zero  
[user@host ~]$ ls -ld zero  
drwxrwxrwx. 2 user user 0 May  9 01:54 zero
```

To mask all file and directory permissions for other, set the umask value to 007.

```
[user@host ~]$ umask 007
[user@host ~]$ touch seven.txt
[user@host ~]$ ls -l seven.txt
-rw-rw----. 1 user user 0 May  9 01:55 seven.txt
[user@host ~]$ mkdir seven
[user@host ~]$ ls -ld seven
drwxrwx---. 2 user user 0 May  9 01:54 seven
```

A umask of 027 ensures that new files have read and write permissions for user and read permission for group. New directories have read and write access for group and no permissions for other.

```
[user@host ~]$ umask 027
[user@host ~]$ touch two-seven.txt
[user@host ~]$ ls -l two-seven.txt
-rw-r-----. 1 user user 0 May  9 01:55 two-seven.txt
[user@host ~]$ mkdir two-seven
[user@host ~]$ ls -ld two-seven
drwxr-x---. 2 user user 0 May  9 01:54 two-seven
```

The default umask for users is set by the shell startup scripts. By default, if your account's UID is 200 or more and your username and primary group name are the same, you will be assigned a umask of 002. Otherwise, your umask will be 022.

As **root**, you can change this by adding a shell startup script named **/etc/profile.d/local-umask.sh** that looks something like the output in this example:

```
[root@host ~]# cat /etc/profile.d/local-umask.sh
# Overrides default umask configuration
if [ $UID -gt 199 ] && [ "`id -gn`" = "`id -un`" ]; then
    umask 007
else
    umask 022
fi
```

The preceding example will set the umask to 007 for users with a UID greater than 199 and with a username and primary group name that match, and to 022 for everyone else. If you just wanted to set the umask for everyone to 022, you could create that file with just the following content:

```
# Overrides default umask configuration
umask 022
```

To ensure that global umask changes take effect you must log out of the shell and log back in. Until that time the umask configured in the current shell is still in effect.



References

bash(1), **ls(1)**, **chmod(1)**, and **umask(1)** man pages

► Guided Exercise

Managing Default Permissions and File Access

In this exercise, you will control the permissions on new files created in a directory by using umask settings and the setgid permission.

Outcomes

You should be able to:

- Create a shared directory where new files are automatically owned by the **operators** group.
- Experiment with various umask settings.
- Adjust default permissions for specific users.
- Confirm your adjustment is correct.

Before You Begin

Log in to **workstation** as **student** using **student** as the password.

On **workstation**, run the **lab perms-default start** command. The command runs a start script that determines if **servera** is reachable on the network. The script also creates the **operators** group and the **operator1** user on **servera**.

```
[student@workstation ~]$ lab perms-default start
```

- 1. Use the **ssh** command to log in to **servera** as the **student** user.

```
[student@workstation ~]$ ssh student@servera  
...output omitted...  
[student@servera ~]$
```

- 2. Use the **su** command to switch to the **operator1** user using **redhat** as the password.

```
[student@servera ~]$ su - operator1  
Password: redhat  
[operator1@servera ~]$
```

- 3. Use the **umask** command to list the **operator1** user's default umask value.

```
[operator1@servera ~]$ umask  
0002
```

- ▶ 4. Create a new directory named **/tmp/shared**. In the **/tmp/shared** directory, create a file named **defaults**. Look at the default permissions.

- 4.1. Use the **mkdir** command to create the **/tmp/shared** directory. Use the **ls -ld** command to list the permissions of the new directory.

```
[operator1@servera ~]$ mkdir /tmp/shared  
[operator1@servera ~]$ ls -ld /tmp/shared  
drwxrwxr-x. 2 operator1 operator1 6 Feb 4 14:06 /tmp/shared
```

- 4.2. Use the **touch** command to create a file named **defaults** in the **/tmp/shared** directory.

```
[operator1@servera ~]$ touch /tmp/shared/defaults
```

- 4.3. Use the **ls -l** command to list the permissions of the new file.

```
[operator1@servera ~]$ ls -l /tmp/shared/defaults  
-rw-rw-r--. 1 operator1 operator1 0 Feb 4 14:09 /tmp/shared/defaults
```

- ▶ 5. Change the group ownership of **/tmp/shared** to **operators**. Confirm the new ownership and permissions.

- 5.1. Use the **chown** command to change the group ownership of the **/tmp/shared** directory to **operators**.

```
[operator1@servera ~]$ chown :operators /tmp/shared
```

- 5.2. Use the **ls -ld** command to list the permissions of the **/tmp/shared** directory.

```
[operator1@servera ~]$ ls -ld /tmp/shared  
drwxrwxr-x. 2 operator1 operators 22 Feb 4 14:09 /tmp/shared
```

- 5.3. Use the **touch** command to create a file named **group** in the **/tmp/shared** directory. Use the **ls -l** command to list the file permissions.

```
[operator1@servera ~]$ touch /tmp/shared/group  
[operator1@servera ~]$ ls -l /tmp/shared/group  
-rw-rw-r--. 1 operator1 operator1 0 Feb 4 17:00 /tmp/shared/group
```



Note

The group owner of the **/tmp/shared/group** file is not **operators** but **operator1**.

- ▶ 6. Ensure that files created in the **/tmp/shared** directory are owned by the **operators** group.

- 6.1. Use the **chmod** command to set the group ID to the **operators** group for the **/tmp/shared** directory.

```
[operator1@servera ~]$ chmod g+s /tmp/shared
```

- 6.2. Use the **touch** command to create a new file named **operations_database.txt** in the **/tmp/shared** directory.

```
[operator1@servera ~]$ touch /tmp/shared/operations_database.txt
```

- 6.3. Use the **ls -l** command to verify that the **operators** group is the group owner for the new file.

```
[operator1@servera ~]$ ls -l /tmp/shared/operations_database.txt
-rw-rw-r--. 1 operator1 operators 0 Feb  4 16:11 /tmp/shared/
operations_database.txt
```

- ▶ 7. Create a new file in the **/tmp/shared** directory named **operations_network.txt**. Record the ownership and permissions. Change the **umask** for **operator1**. Create a new file called **operations_production.txt**. Record the ownership and permissions of the **operations_production.txt** file.
- 7.1. Use the **touch** command to create a file called **operations_network.txt** in the **/tmp/shared** directory.

```
[operator1@servera ~]$ touch /tmp/shared/operations_network.txt
```

- 7.2. Use the **ls -l** command to list the permissions of the **operations_network.txt** file.

```
[operator1@servera ~]$ ls -l /tmp/shared/operations_network.txt
-rw-rw-r--. 1 operator1 operators 5 Feb  0 15:43 /tmp/shared/
operations_network.txt
```

- 7.3. Use the **umask** command to change the umask for the **operator1** user to 027. Use the **umask** command to confirm the change.

```
[operator1@servera ~]$ umask 027
[operator1@servera ~]$ umask
0027
```

- 7.4. Use the **touch** command to create a new file named **operations_production.txt** in the **/tmp/shared/** directory. Use the **ls -l** command to ensure that newly created files are created with read-only access for the **operators** group and no access for other users.

```
[operator1@servera ~]$ touch /tmp/shared/operations_production.txt
[operator1@servera ~]$ ls -l /tmp/shared/operations_production.txt
-rw-r-----. 1 operator1 operators 0 Feb  0 15:56 /tmp/shared/
operations_production.txt
```

- ▶ 8. Open a new terminal window and log in to **servera** as **operator1**.

```
[student@workstation ~]$ ssh operator1@servera  
...output omitted...  
[operator1@servera ~]$
```

- 9. List the umask value for **operator1**.

```
[operator1@servera ~]$ umask  
0002
```

- 10. Change the default umask for the **operator1** user. The new umask prohibits all access for users not in their group. Confirm that the umask has been changed.

- 10.1. Use the **echo** command to change the default umask for the **operator1** user to 007.

```
[operator1@servera ~]$ echo "umask 007" >> ~/.bashrc  
[operator1@servera ~]$ 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
```

- 10.2. Log out and log in again as the **operator1** user. Use the **umask** command to confirm that the change is permanent.

```
[operator1@servera ~]$ exit  
logout  
Connection to servera closed.  
[student@workstation ~]$ ssh operator1@servera  
...output omitted...  
[operator1@servera ~]$ umask  
0007
```

- 11. On **servera**, exit from all the **operator1** and the **student** user shells.



Warning

Exit from all shells opened by **operator1**. Failure to exit from all shells will cause the finish script to fail.

```
[operator1@servera ~]$ exit  
logout  
Connection to servera closed.  
[student@workstation ~]$
```

Finish

On **workstation**, run the **lab perms-default finish** script to complete this exercise.

```
[student@workstation ~]$ lab perms-default finish
```

This concludes the guided exercise.

► Lab

Controlling Access to Files

Performance Checklist

In this lab, you will configure permissions on files and set up a directory that users in a particular group can use to conveniently share files on the local file system.

Outcomes

You should be able to:

- Create a directory where users can work collaboratively on files.
- Create files that are automatically assigned group ownership.
- Create files that are not accessible outside of the group.

Before You Begin

Log in to **workstation** as **student** using **student** as the password.

On **workstation**, run the **lab perms-review start** command. The command runs a start script that determines if **serverb** is reachable on the network. The script also creates the **techdocs** group and three users named **tech1**, **tech2**, and **database1**.

```
[student@workstation ~]$ lab perms-review start
```

1. Use the **ssh** command to log in to **serverb** as the **student** user. Switch to **root** on **serverb** using **redhat** as the password.
2. Create a directory called **/home/techdocs**.
3. Change the group ownership of the **/home/techdocs** directory to the **techdocs** group.
4. Verify that users in the **techdocs** group cannot currently create files in the **/home/techdocs** directory.
5. Set permissions on the **/home/techdocs** directory. On the **/home/techdocs** directory, configure setgid (2), read/write/execute permissions (7) for the owner/user and group, and no permissions (0) for other users.
6. Verify that the permissions are set properly.
7. Confirm that users in the **techdocs** group can now create and edit files in the **/home/techdocs** directory. Users not in the **techdocs** group cannot edit or create files in the **/home/techdocs** directory. Users **tech1** and **tech2** are in the **techdocs** group. User **database1** is not in that group.
8. Modify the global login scripts. Normal users should have a umask setting that prevents others from viewing or modifying new files and directories.
9. Log off from **serverb**.

```
[student@serverb ~]$ exit  
logout  
Connection to serverb closed.
```

Evaluation

On **workstation**, run the **lab perms-review grade** script to confirm success on this exercise.

```
[student@workstation ~]$ lab perms-review grade
```

Finish

On **workstation**, run the **lab perms-review finish** script to complete the lab.

```
[student@workstation ~]$ lab perms-review finish
```

This concludes the lab.

► Solution

Controlling Access to Files

Performance Checklist

In this lab, you will configure permissions on files and set up a directory that users in a particular group can use to conveniently share files on the local file system.

Outcomes

You should be able to:

- Create a directory where users can work collaboratively on files.
- Create files that are automatically assigned group ownership.
- Create files that are not accessible outside of the group.

Before You Begin

Log in to **workstation** as **student** using **student** as the password.

On **workstation**, run the **lab perms-review start** command. The command runs a start script that determines if **serverb** is reachable on the network. The script also creates the **techdocs** group and three users named **tech1**, **tech2**, and **database1**.

```
[student@workstation ~]$ lab perms-review start
```

1. Use the **ssh** command to log in to **serverb** as the **student** user. Switch to **root** on **serverb** using **redhat** as the password.

```
[student@workstation ~]$ ssh student@serverb
...output omitted...
[student@serverb ~]$ su -
Password: redhat
[root@serverb ~]#
```

2. Create a directory called **/home/techdocs**.
 - 2.1. Use the **mkdir** command to create a directory called **/home/techdocs**.

```
[root@serverb ~]# mkdir /home/techdocs
```

3. Change the group ownership of the **/home/techdocs** directory to the **techdocs** group.
 - 3.1. Use the **chown** command to change the group ownership for the **/home/techdocs** directory to the **techdocs** group.

```
[root@serverb ~]# chown :techdocs /home/techdocs
```

4. Verify that users in the **techdocs** group cannot currently create files in the **/home/techdocs** directory.

4.1. Use the **su** command to switch to the **tech1** user.

```
[root@serverb ~]# su - tech1  
[tech1@serverb ~]$
```

4.2. Use **touch** to create a file named **techdoc1.txt** in the **/home/techdocs** directory.

```
[tech1@serverb ~]$ touch /home/techdocs/techdoc1.txt  
touch: cannot touch '/home/techdocs/techdoc1.txt': Permission denied
```



Note

Note that even though the **/home/techdocs** directory is owned by **techdocs** and **tech1** is part of the **techdocs** group, it is not possible to create a new file in that directory. This is because the **techdocs** group does not have write permission. Use the **ls -ld** command to show the permissions.

```
[tech1@serverb ~]$ ls -ld /home/techdocs/  
drwxr-xr-x. 2 root techdocs 6 Feb 5 16:05 /home/techdocs/
```

5. Set permissions on the **/home/techdocs** directory. On the **/home/techdocs** directory, configure setgid (2), read/write/execute permissions (7) for the owner/user and group, and no permissions (0) for other users.

5.1. Exit from the **tech1** user shell.

```
[tech1@serverb ~]$ exit  
logout  
[root@serverb ~]#
```

5.2. Use the **chmod** command to set the group permission for the **/home/techdocs** directory. On the **/home/techdocs** directory, configure setgid (2), read/write/execute permissions (7) for the owner/user and group, and no permissions (0) for other users.

```
[root@serverb ~]$ chmod 2770 /home/techdocs
```

6. Verify that the permissions are set properly.

```
[root@serverb ~]$ ls -ld /home/techdocs  
drwxrws---. 2 root techdocs 6 Feb 4 18:12 /home/techdocs/
```

Note that the **techdocs** group now has write permission.

7. Confirm that users in the **techdocs** group can now create and edit files in the **/home/techdocs** directory. Users not in the **techdocs** group cannot edit or create files in the **/home/techdocs** directory. Users **tech1** and **tech2** are in the **techdocs** group. User **database1** is not in that group.

- 7.1. Switch to the **tech1** user. Use **touch** to create a file called **techdoc1.txt** in the **/home/techdocs** directory. Exit from the **tech1** user shell.

```
[root@serverb ~]# su - tech1
[tech1@serverb ~]$ touch /home/techdocs/techdoc1.txt
[tech1@serverb ~]$ ls -l /home/techdocs/techdoc1.txt
-rw-rw-r--. 1 tech1 techdocs 0 Feb  5 16:42 /home/techdocs/techdoc1.txt
[tech1@serverb ~]$ exit
logout
[root@serverb ~]#
```

- 7.2. Switch to the **tech2** user. Use the **echo** command to add some content to the **/home/techdocs/techdoc1.txt** file. Exit from the **tech2** user shell.

```
[root@serverb ~]# su - tech2
[tech2@serverb ~]$ cd /home/techdocs
[tech2@serverb techdocs]$ echo "This is the first tech doc." > techdoc1.txt
[tech2@serverb techdocs]$ exit
logout
[root@serverb ~]#
```

- 7.3. Switch to the **database1** user. Use the **echo** command to append some content to the **/home/techdocs/techdoc1.txt** file. Notice that you will get a **Permission Denied** message. Use the **ls -l** command to confirm that **database1** does not have access to the file. Exit from the **database1** user shell.

The following **echo** command is very long and should be entered on a single line.

```
[root@serverb ~]# su - database1
[database1@serverb ~]$ echo "This is the first tech doc." >> /home/techdocs/
techdoc1.txt
-bash: /home/techdocs/techdoc1.txt: Permission denied
[database1@serverb ~]$ ls -l /home/techdocs/techdoc1.txt
ls: cannot access '/home/techdocs/techdoc1.txt': Permission denied
[database1@serverb ~]$ exit
logout
[root@serverb ~]#
```

8. Modify the global login scripts. Normal users should have a umask setting that prevents others from viewing or modifying new files and directories.

- 8.1. Determine the umask of the **student** user. Use the **su - student** command to switch to **student** login shell. When done exit from the shell.

```
[root@serverb ~]# su - student
[student@serverb ~]$ umask
0002
[student@serverb ~]$ exit
logout
[root@serverb ~]#
```

- 8.2. Create the **/etc/profile.d/local-umask.sh** file with the following content to set the umask to **007** for users with a UID greater than **199** and with a username and primary group name that match, and to **022** for everyone else:

```
# Overrides default umask configuration
if [ $UID -gt 199 ] && [ "`id -gn`" = "`id -un`" ]; then
    umask 007
else
    umask 022
fi
```

- 8.3. Log out of the shell and log back in as **student** to verify that global umask changes to **007**.

```
[root@serverb ~]# exit
logout
[student@serverb ~]$ exit
logout
Connection to serverb closed.
[student@workstation ~]$ ssh student@serverb
...output omitted...
[student@serverb ~]$ umask
0007
```

9. Log off from **serverb**.

```
[student@serverb ~]$ exit
logout
Connection to serverb closed.
```

Evaluation

On **workstation**, run the **lab perms-review grade** script to confirm success on this exercise.

```
[student@workstation ~]$ lab perms-review grade
```

Finish

On **workstation**, run the **lab perms-review finish** script to complete the lab.

```
[student@workstation ~]$ lab perms-review finish
```

This concludes the lab.

Summary

In this chapter, you learned:

- Files have three categories to which permissions apply. A file is owned by a user, a single group, and other users. The most specific permission applies. User permissions override group permissions and group permissions override other permissions.
- The **ls** command with the **-l** option expands the file listing to include both the file permissions and ownership.
- The **chmod** command changes file permissions from the command line. There are two methods to represent permissions, symbolic (letters) and numeric (digits).
- The **chown** command changes file ownership. The **-R** option recursively changes the ownership of a directory tree.
- The **umask** command without arguments displays the current umask value of the shell. Every process on the system has a umask. The default umask values for Bash are defined in the **/etc/profile** and **/etc/bashrc** files.

Chapter 8

Monitoring and Managing Linux Processes

Goal

Evaluate and control processes running on a Red Hat Enterprise Linux system.

Objectives

- Get information about programs running on the system so that you can determine status, resource use, and ownership, so you can control them.
- Use Bash job control to manage multiple processes started from the same terminal session.
- Control and terminate processes that are not associated with your shell, and forcibly end user sessions and processes.
- Describe what load average is and determine processes responsible for high resource use on a server.

Sections

- Listing Processes (and Quiz)
- Controlling Jobs (and Guided Exercise)
- Killing Processes (and Guided Exercise)
- Monitoring Process Activity (and Guided Exercise)

Lab

Monitoring and Managing Linux Processes

Listing Processes

Objectives

After completing this section, you should be able to get information about programs running on a system to determine status, resource use, and ownership, so you can control them.

Definition of 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
- Process state

The *environment* of a process includes:

- Local and global variables
- A current scheduling context
- 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, **systemd** on a Red Hat Enterprise Linux 8 system).

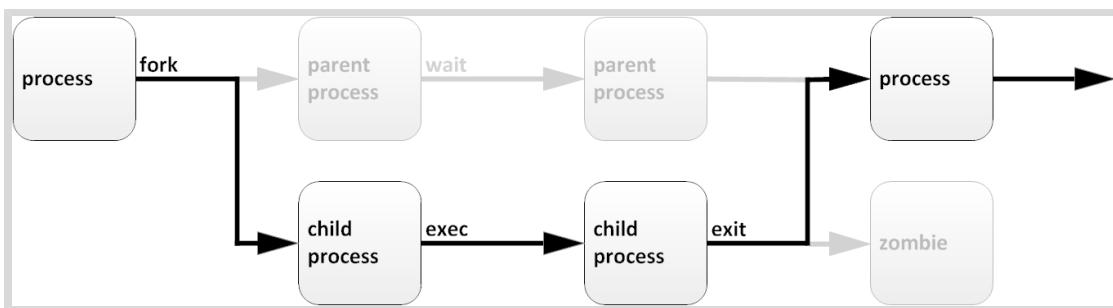


Figure 8.1: Process life cycle

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 only remaining resource, called a *zombie*, is an entry in the process table. The parent, signaled awake when the child exited, cleans the process table of the child's entry, thus freeing the last resource of the child process. The parent process then continues with its own program code execution.

Describing 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 dictate.

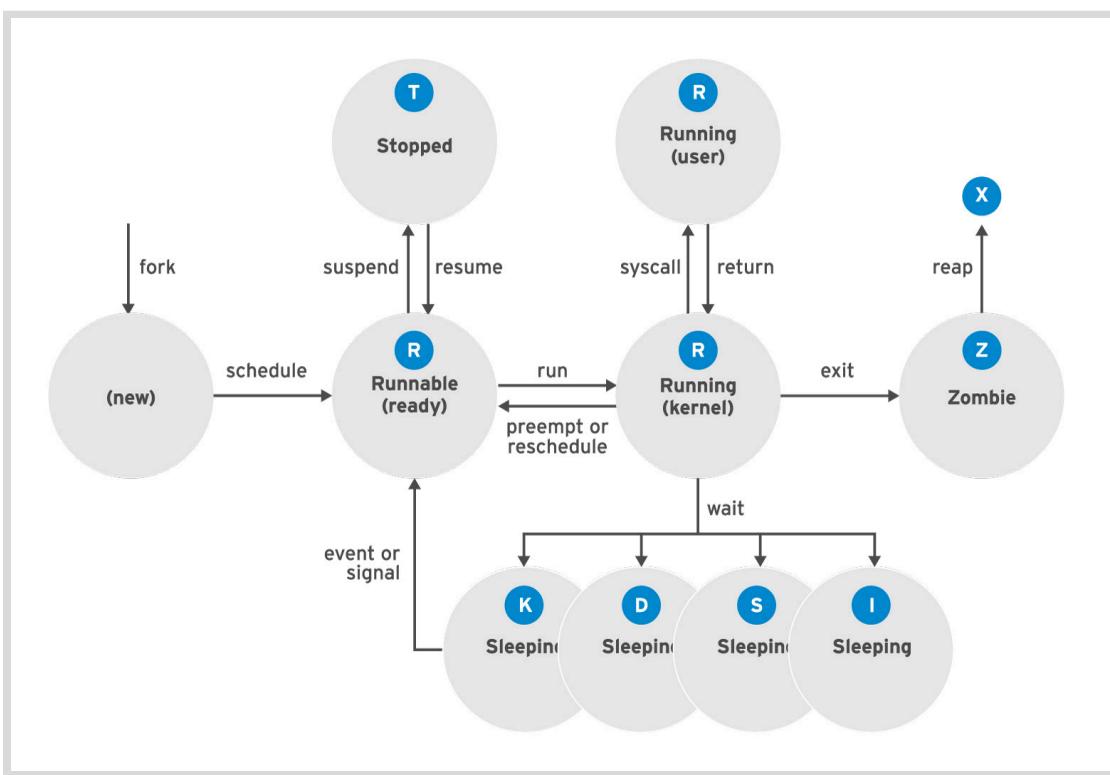


Figure 8.2: Linux process states

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, does not respond to signals. Used only when process interruption may cause an unpredictable device state.
	K	TASK_KILLABLE: Identical to the uninterruptible D state, but modified to allow a waiting task to respond to the signal that it should be killed (exit completely). Utilities frequently display <i>Killable</i> processes as D state.

Name	Flag	Kernel-defined state name and description
	I	TASK_REPORT_IDLE: A subset of state D. The kernel does not count these processes when calculating load average. Used for kernel threads. Flags TASK_UNINTERRUPTABLE and TASK_NOLOAD are set. Similar to TASK_KILLABLE, also a subset of state D. It accepts fatal signals.
Stopped	T	TASK_STOPPED: The process has been Stopped (suspended), usually by being signaled by a user or another process. The process can be continued (resumed) by another signal to return to Running.
	T	TASK_TRACED: A process that is being debugged is also temporarily Stopped 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.

Why Process States are Important

When troubleshooting a system, it is important to understand how the kernel communicates with processes and how processes communicate with each other. At process creation, the system assigns the process a state. The **S** column of the **top** command or the **STAT** column of the **ps** show the state of each process. On a single CPU system, only one process can run at a time. It is possible to see several processes with a state of **R**. However, not all of them will be running consecutively, some of them will be in status *waiting*.

```
[user@host ~]$ top
 PID USER PR NI    VIRT    RES    SHR S %CPU %MEM   TIME+ COMMAND
  1 root  20  0 244344 13684  9024 S  0.0  0.7  0:02.46 systemd
  2 root  20  0      0      0      0 S  0.0  0.0  0:00.00 kthreadd
...output omitted...
```

```
[user@host ~]$ ps aux
USER      PID %CPU %MEM      VSZ      RSS TTY      STAT START   TIME COMMAND
...output omitted...
root        2  0.0  0.0      0      0 ?      S    11:57   0:00 [kthreadd]
student   3448  0.0  0.2 266904  3836 pts/0     R+   18:07   0:00 ps aux
...output omitted...
```

Process can be suspended, stopped, resumed, terminated, and interrupted using signals. Signals are discussed in more detail later in this chapter. Signals can be used by other processes, by the kernel itself, or by users logged into the system.

Listing Processes

The **ps** command is used for listing current processes. It can provide detailed process information, including:

- User identification (UID), which determines process privileges

- Unique process identification (PID)
- 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*
- The current process state



Important

The Linux version of **ps** supports three option formats:

- 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
- GNU long options, which are preceded by two dashes

For example, **ps -aux** is not the same as **ps aux**.

Perhaps the most common set of options, **aux**, displays all processes including processes without a controlling terminal. A long listing (options **lax**) provides more technical detail, but may display faster by avoiding user name lookups. The similar UNIX syntax uses the options **-ef** to display all processes.

```
[user@host ~]$ 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/
sys
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 omitted...
[user@host ~]$ ps lax
F  UID  PID  PPIID 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      0 kthrea S   ?      0:00 [kthreadd]
1  0     3      2  20   0      0      0 smpboo S   ?      0:00 [ksoftirqd/0]
1  0     5      2  0 -20    0      0 worker S<  ?      0:00 [kworker/0:0H]
1  0     7      2 -100  -    0      0 smpboo S   ?      0:00 [migration/0]
...output omitted...
[user@host ~]$ ps -ef
UID      PID  PPIID C STIME TTY      TIME CMD
root      1      0  0 17:45 ?
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 omitted...
```

By default, **ps** with no options selects all processes with the same *effective user ID* (EUID) as the current user, and which are associated with the same terminal where **ps** was invoked.

- Processes in brackets (usually at the top of the list) are scheduled kernel threads.
- Zombies are listed as **exiting** or **defunct**.
- The output of **ps** displays once. Use **top** for a process display that dynamically updates.
- **ps** can display in tree format so you can view relationships between parent and child processes.
- The default output is sorted by process ID number. At first glance, this may appear to be chronological order. However, the kernel reuses process IDs, so the order is less structured than it appears. To sort, use the **-o** or **--sort** options. 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

► Quiz

Listing Processes

Choose the correct answers to the following questions:

► 1. Which state represents a process that has been stopped or suspended?

- a. D
- b. R
- c. S
- d. T
- e. Z

► 2. Which state represents a process that has released all of its resources except its PID?

- a. D
- b. R
- c. S
- d. T
- e. Z

► 3. Which process does a parent use to duplicate to create a new child process?

- a. exec
- b. fork
- c. zombie
- d. syscall
- e. reap

► 4. Which state represents a process that is sleeping until some condition is met?

- a. D
- b. R
- c. S
- d. T
- e. Z

► Solution

Listing Processes

Choose the correct answers to the following questions:

► 1. Which state represents a process that has been stopped or suspended?

- a. D
- b. R
- c. S
- d. T
- e. Z

► 2. Which state represents a process that has released all of its resources except its PID?

- a. D
- b. R
- c. S
- d. T
- e. Z

► 3. Which process does a parent use to duplicate to create a new child process?

- a. exec
- b. fork
- c. zombie
- d. syscall
- e. reap

► 4. Which state represents a process that is sleeping until some condition is met?

- a. D
- b. R
- c. S
- d. T
- e. Z

Controlling Jobs

Objectives

After completing this section, you should be able to use Bash job control to manage multiple processes started from the same terminal session.

Describing Jobs and Sessions

Job control is a feature of the shell which allows a single shell instance to run and manage multiple commands.

A *job* is associated with each pipeline entered at a shell prompt. All processes in that pipeline are part of the job and are members of the same *process group*. If only one command is entered at a shell prompt, that can be considered to be a minimal “pipeline” of one command, creating a job with only one member.

Only one job can read input and keyboard generated signals from a particular terminal window at a time. Processes that are part of that job are *foreground* processes of that *controlling terminal*.

A *background* process of that controlling terminal is a member of any other job associated with that terminal. Background processes of a terminal cannot read input or receive keyboard generated interrupts from the terminal, but may be able to write to the terminal. A job in the background may be stopped (suspended) or it may be running. If a running background job tries to read from the terminal, it will be automatically suspended.

Each terminal is its own *session*, and can have a foreground process and any number of independent background processes. A job is part of exactly one session: the one belonging to its controlling terminal.

The **ps** command shows the device name of the controlling terminal of a process in the **TTY** column. Some processes, such as *system daemons*, are started by the system and not from a shell prompt. These processes do not have a controlling terminal, are not members of a job, and cannot be brought to the foreground. The **ps** command displays a question mark (?) in the **TTY** column for these processes.

Running Jobs in the Background

Any command or pipeline can be started in the background by appending an ampersand (&) to the end of the command line. The Bash shell displays a *job number* (unique to the session) and the PID of the new child process. The shell does not wait for the child process to terminate, but rather displays the shell prompt.

```
[user@host ~]$ sleep 10000 &
[1] 5947
[user@host ~]$
```

**Note**

When a command line containing a pipe is sent to the background using an ampersand, the PID of the last command in the pipeline is used as output. All processes in the pipeline are still members of that job.

```
[user@host ~]$ example_command | sort | mail -s "Sort output" &
[1] 5998
```

You can display the list of jobs that Bash is tracking for a particular session with the **jobs** command.

```
[user@host ~]$ jobs
[1]+  Running                  sleep 10000 &
[user@host ~]$
```

A background job can be brought to the foreground by using the **fg** command with its job ID (%job number).

```
[user@host ~]$ fg %1
sleep 10000
```

In the preceding example, the **sleep** command is now running in the foreground on the controlling terminal. The shell itself is again asleep, waiting for this child process to exit.

To send a foreground process to the background, first press the keyboard generated *suspend request* (**Ctrl+z**) in the terminal.

```
sleep 10000
^Z
[1]+  Stopped                  sleep 10000
[user@host ~]$
```

The job is immediately placed in the background and is suspended.

The **ps j** command displays information relating to jobs. The PID is the unique *process ID* of the process. The PPID is the PID of the *parent process* of this process, the process that started (forked) it. The PGID is the PID of the *process group leader*, normally the first process in the job's pipeline. The SID is the PID of the *session leader*, which (for a job) is normally the interactive shell that is running on its controlling terminal. Since the example **sleep** command is currently suspended, its process state is **T**.

```
[user@host ~]$ 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
```

To start the suspended process running in the background, use the **bg** command with the same job ID.

```
[user@host ~]$ bg %1  
[1]+ sleep 10000 &
```

The 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.



Note

Note the **+** sign after the **[1]** in the examples above. The **+** sign indicates that this job is the current default job. That is, if a command is used that expects a **%job number** argument and a job number is not provided, then the action is taken on the job with the **+** indicator.



References

Bash info page (*The GNU Bash Reference Manual*)
<https://www.gnu.org/software/bash/manual>

- Section 7: Job Control

bash(1), builtins(1), ps(1), sleep(1) man pages

► Guided Exercise

Controlling Jobs

In this exercise, you will start, suspend, background, and foreground multiple processes using job control.

Outcomes

You should be able to use job control to suspend and restart user processes.

Before You Begin

Log in to **workstation** as **student** using **student** as the password.

On **workstation**, run the **lab processes-control start** command. This script ensures that **servera** is available.

```
[student@workstation ~]$ lab processes-control start
```

- ▶ 1. On **workstation**, open two terminal windows side by side. In this section, these two terminals are referred to as *left* and *right*. In each terminal, use the **ssh** command to log in to **servera** as the **student** user.

```
[student@workstation ~]$ ssh student@servera
...output omitted...
[student@servera ~]$
```

- ▶ 2. In the *left* window, create a new directory called **/home/student/bin**. In the new directory, create a shell script called **control**. Make the script executable.

- 2.1. Use the **mkdir** command to create a new directory called **/home/student/bin**.

```
[student@servera ~]$ mkdir /home/student/bin
```

- 2.2. Use the **vim** command to create a script called **control** in the **/home/student/bin** directory. To enter Vim interactive mode, press the **i** key. Use the **:wq** command to save the file.

```
[student@servera ~]$ vim /home/student/bin/control
#!/bin/bash
while true; do
    echo -n "$@" >> ~/control_outfile
    sleep 1
done
```

**Note**

The `control` script runs until terminated. It appends command-line arguments to the file `~/control_outfile` once per second.

- 2.3. Use the `chmod` command to make the `control` file executable.

```
[student@servera ~]$ chmod +x /home/student/bin/control
```

- ▶ 3. Execute the `control` script. The script continuously appends the word "technical" and a space to the file `~/control_outfile` at one second intervals.

**Note**

You are able to execute your `control` script because it is located in your `PATH`, and has been made executable.

```
[student@servera ~]$ control technical
```

- ▶ 4. In the right terminal shell, use the `tail` command with the `-f` option to confirm that the new process is writing to the `/home/student/control_outfile` file.

```
[student@servera ~]$ tail -f ~/control_outfile
technical technical technical technical
...output omitted...
```

- ▶ 5. In the left terminal shell, press `Ctrl+z` to suspend the running process. The shell returns the job ID in square brackets. In the right window, confirm that the process output has stopped.

```
^Z
[1]+  Stopped                  control technical
[student@servera ~]$
```

```
technical technical technical technical
...no further output...
```

- ▶ 6. In the left terminal shell, view the `jobs` list. Remember that the `+` sign indicates the default job. Restart the job in the background. In the right terminal shell, confirm that the process output is again active.

- 6.1. Using the `jobs` command, view the list of jobs.

```
[student@servera ~]$ jobs
[1]+  Stopped                  control technical
```

- 6.2. Using the `bg` command, restart the `control` job in the background.

```
[student@servera ~]$ bg
[1]+ control technical &
```

- 6.3. Use the **jobs** command to confirm that the **control** job is running again.

```
[student@servera ~]$ jobs
[1]+ Running control technical &
```

- 6.4. In the right terminal shell, confirm that the **tail** command is producing output.

```
...output omitted...
technical technical technical technical technical technical technical technical
```

- 7. In the left terminal shell, start two more **control** processes to append to the **~/output** file. Use the ampersand (&) to start the processes in the background. Replace **technical** with **documents** and then with **database**. Replacing the arguments helps to differentiate between the three processes.

```
[student@servera ~]$ control documents &
[2] 6579
[student@servera ~]$
[student@servera ~]$ control database &
[3] 6654
```



Note

The job number of each new process is printed in square brackets. The second number is the unique system-wide process ID number (PID) for the process.

- 8. In the left terminal shell, use the **jobs** command to view the three running processes. In the right terminal shell, confirm that all three processes are appending to the file.

```
[student@servera ~]$ jobs
[1] Running control technical &
[2]- Running control documents &
[3]+ Running control database &
```

```
...output omitted...
technical documents database technical documents database technical documents
database technical documents database
...output omitted...
```

- 9. Suspend the **control technical** process. Confirm that it has been suspended. Terminate the **control documents** process and confirm that it has been terminated.

- 9.1. In the left terminal shell, use the **fg** command with the job ID to foreground the **control technical** process. Press **Ctrl+z** to suspend the process. Use the **jobs** command to confirm that the process is suspended.

```
[student@servera ~]$ fg %1
control technical
^Z
[1]+  Stopped                  control technical
[student@servera ~]$ jobs
[1]+  Stopped                  control technical
[2]  Running                   control documents &
[3]-  Running                  control database &
```

- 9.2. In the right terminal shell, confirm that the **control technical** process is no longer sending output.

```
database documents database documents database
...no further output...
```

- 9.3. In the left terminal shell, use the **fg** command with the job ID to foreground the **control documents** process. Press **Ctrl+c** to terminate the process. Use the **jobs** command to confirm that the process is terminated.

```
[student@servera ~]$ fg %2
control documents
^C
[student@servera ~]$ jobs
[1]+  Stopped                  control technical
[3]-  Running                  control database &
```

- 9.4. In the right terminal shell, confirm that the **control documents** process is no longer sending output.

```
...output omitted...
database database database database database database database
...no further output...
```

- 10. In the left window, use the **ps** command with the **jT** option to view the remaining jobs. The suspended jobs have a state of **T**. The other background jobs are sleeping (**S**).

```
[student@servera ~]$ ps jT
  PID  PID  PGID  SID TTY      TPGID STAT   UID    TIME COMMAND
27277 27278 27278 27278 pts/1    28702 Ss    1000   0:00 -bash
27278 28234 28234 27278 pts/1    28702 T     1000   0:00 /bin/bash /home/student/
bin/control technical
27278 28251 28251 27278 pts/1    28702 S     1000   0:00 /bin/bash /home/student/
bin/control database
28234 28316 28234 27278 pts/1    28702 T     1000   0:00 sleep 1
28251 28701 28251 27278 pts/1    28702 S     1000   0:00 sleep 1
27278 28702 28702 27278 pts/1    28702 R+    1000   0:00 ps jT
```

- 11. In the left window, use the **jobs** command to view the current jobs. Terminate the **control database** process and confirm that it has been terminated.

```
[student@servera ~]$ jobs  
[1]+  Stopped                  control technical  
[3]-  Running                  control database &
```

Use the **fg** command with the job ID to foreground the **control database** process. Press **Ctrl+c** to terminate the process. Use the **jobs** command to confirm that the process is terminated.

```
[student@servera ~]$ fg %3  
control database  
^C  
[student@servera ~]$ jobs  
[1]+  Stopped                  control technical
```

- ▶ 12. In the right terminal shell, use the **Ctrl+c** command to stop the **tail** command. Using the **rm** command, delete the **~/control_outfile** file.

```
...output omitted...  
Ctrl+c  
[student@servera ~]$ rm ~/control_outfile
```

- ▶ 13. Log out from **servera** on both terminals.

```
[student@servera ~]$ exit  
logout  
Connection to servera closed.
```

```
[student@servera ~]$ exit  
logout  
Connection to servera closed.
```

Finish

On **workstation**, run the **lab processes-control finish** script to complete this exercise.

```
[student@workstation ~]$ lab processes-control finish
```

This concludes the guided exercise.

Killing Processes

Objectives

After completing this section, you 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 (an I/O request or an expired timer), or by explicit use of a signal-sending command or 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 pressing INTR key sequence (Ctrl+c).
3	QUIT	Keyboard quit	Similar to SIGINT; adds a process dump at termination. Sent by pressing QUIT key sequence (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-clean-up.
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 pressing SUSP key sequence (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 x86_64 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 to react to expected event signals by implementing handler routines to ignore, replace, or extend a signal's default action.

Commands for Sending Signals by Explicit Request

You signal the current foreground process by pressing a keyboard control sequence to suspend (**Ctrl+z**), kill (**Ctrl+c**), or core dump (**Ctrl+**) the process. However, you will use signal-sending commands to send signals to a background process or to processes in a different session.

Signals can be specified as options either by name (for example, **-HUP** or **-SIGHUP**) or by number (the related **-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 PID number. Despite its name, the **kill** command can be used to send any signal, not just those for terminating programs. You can use the **kill -l** command to list the names and numbers of all available signals.

```
[user@host ~]$ kill -l
 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 omitted...
[user@host ~]$ ps aux | grep job
5194  0.0  0.1 222448  2980 pts/1    S    16:39   0:00 /bin/bash /home/user/bin/
control job1
5199  0.0  0.1 222448  3132 pts/1    S    16:39   0:00 /bin/bash /home/user/bin/
control job2
5205  0.0  0.1 222448  3124 pts/1    S    16:39   0:00 /bin/bash /home/user/bin/
control job3
5430  0.0  0.0 221860  1096 pts/1    S+   16:41   0:00 grep --color=auto job
[user@host ~]$ kill 5194
[user@host ~]$ ps aux | grep job
user  5199  0.0  0.1 222448  3132 pts/1    S    16:39   0:00 /bin/bash /home/
user/bin/control job2
user  5205  0.0  0.1 222448  3124 pts/1    S    16:39   0:00 /bin/bash /home/
user/bin/control job3
user  5783  0.0  0.0 221860   964 pts/1    S+   16:43   0:00 grep --color=auto
job
[1]  Terminated                  control job1
[user@host ~]$ kill -9 5199
```

```
[user@host ~]$ ps aux | grep job
user  5205  0.0  0.1 222448  3124 pts/1      S     16:39   0:00 /bin/bash /home/
user/bin/control job3
user  5930  0.0  0.0 221860  1048 pts/1      S+    16:44   0:00 grep --color=auto
job
[2]-  Killed                  control job2
[user@host ~]$ kill -SIGTERM 5205
user  5986  0.0  0.0 221860  1048 pts/1      S+    16:45   0:00 grep --color=auto
job
[3]+  Terminated              control job3
```

The **killall** command can signal multiple processes, based on their command name.

```
[user@host ~]$ ps aux | grep job
5194  0.0  0.1 222448  2980 pts/1      S     16:39   0:00 /bin/bash /home/user/bin/
control job1
5199  0.0  0.1 222448  3132 pts/1      S     16:39   0:00 /bin/bash /home/user/bin/
control job2
5205  0.0  0.1 222448  3124 pts/1      S     16:39   0:00 /bin/bash /home/user/bin/
control job3
5430  0.0  0.0 221860  1096 pts/1      S+    16:41   0:00 grep --color=auto job
[user@host ~]$ killall control
[1]  Terminated                  control job1
[2]- Terminated                  control job2
[3]+ Terminated                  control job3
[user@host ~]$
```

Use **pkill** to send a signal to one or more processes which match selection criteria. Selection criteria can be a command name, a process owned by a specific user, or all system-wide processes. The **pkill** command includes advanced selection criteria:

- 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.

```
[user@host ~]$ ps aux | grep pkill
user  5992  0.0  0.1 222448  3040 pts/1      S     16:59   0:00 /bin/bash /home/
user/bin/control pkill1
user  5996  0.0  0.1 222448  3048 pts/1      S     16:59   0:00 /bin/bash /home/
user/bin/control pkill2
user  6004  0.0  0.1 222448  3048 pts/1      S     16:59   0:00 /bin/bash /home/
user/bin/control pkill3
[user@host ~]$ pkill control
[1]  Terminated                  control pkill1
[2]- Terminated                  control pkill2
[user@host ~]$ ps aux | grep pkill
user  6219  0.0  0.0 221860  1052 pts/1      S+    17:00   0:00 grep --color=auto
pkill
[3]+  Terminated                  control pkill3
[user@host ~]$ ps aux | grep test
user  6281  0.0  0.1 222448  3012 pts/1      S     17:04   0:00 /bin/bash /home/
user/bin/control test1
```

```

user  6285  0.0  0.1 222448  3128 pts/1      S    17:04   0:00 /bin/bash /home/
user/bin/control test2
user  6292  0.0  0.1 222448  3064 pts/1      S    17:04   0:00 /bin/bash /home/
user/bin/control test3
user  6318  0.0  0.0 221860  1080 pts/1      S+   17:04   0:00 grep --color=auto
test
[user@host ~]$ pkill -U user
[user@host ~]$ ps aux | grep test
user  6870  0.0  0.0 221860  1048 pts/0      S+   17:07   0:00 grep --color=auto
test
[user@host ~]$

```

Logging Users Out Administratively

You may need to log other users off for any of a variety of reasons. To name a few of the many possibilities: the user committed a security violation; the user may have overused resources; the user may have an unresponsive system; or the user has improper access to materials. In these cases, you may need to administratively terminate their session using signals.

To log off a user, first identify the login session to be terminated. Use the **w** command to list user logins and current running processes. Note the **TTY** and **FROM** columns to determine the sessions to close.

All user login sessions are associated with a terminal device (TTY). If the device name is of the form **pts/N**, it is a *pseudo-terminal* associated with a graphical terminal window or remote login session. If it is of the form **ttyN**, the user is on a system console, alternate console, or other directly connected terminal device.

```

[user@host ~]$ w
12:43:06 up 27 min,  5 users,  load average: 0.03,  0.17,  0.66
USER     TTY     FROM           LOGIN@   IDLE   JCPU   PCPU WHAT
root     tty2          12:26   14:58   0.04s  0.04s -bash
bob      tty3          12:28   14:42   0.02s  0.02s -bash
user     pts/1  desk.example.com 12:41   2.00s  0.03s  0.03s w
[user@host ~]$

```

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, including background tasks and child processes, are in the **JCPU** column. Current foreground process CPU consumption is in the **PCPU** column.

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.



Important

SIGKILL is commonly used too quickly by administrators.

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 try SIGINT, and only if both fail retry with SIGKILL.

First identify the PID numbers to be killed using **pgrep**, which operates much like **pkill**, including using the same options, except that **pgrep** lists processes rather than killing them.

```
[root@host ~]# pgrep -l -u bob
6964 bash
6998 sleep
6999 sleep
7000 sleep
[root@host ~]# pkill -SIGKILL -u bob
[root@host ~]# pgrep -l -u bob
[root@host ~]#
```

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 referencing 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@host ~]# pgrep -l -u bob
7391 bash
7426 sleep
7427 sleep
7428 sleep
[root@host ~]# w -h -u bob
bob      tty3      18:37    5:04    0.03s  0.03s -bash
[root@host ~]# pkill -t tty3
[root@host ~]# pgrep -l -u bob
7391 bash
[root@host ~]# pkill -SIGKILL -t tty3
[root@host ~]# pgrep -l -u bob
[root@host ~]#
```

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@host ~]# pstree -p bob
bash(8391)->sleep(8425)
              |-sleep(8426)
              \-sleep(8427)
[root@host ~]# pkill -P 8391
[root@host ~]# pgrep -l -u bob
bash(8391)
[root@host ~]# pkill -SIGKILL -P 8391
[root@host ~]# pgrep -l -u bob
bash(8391)
[root@host ~]#
```



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

► Guided Exercise

Killing Processes

In this exercise, you will use signals to manage and stop processes.

Outcomes

You should be able to start and stop multiple shell processes.

Before You Begin

Log in to **workstation** as **student** using **student** as the password.

On **workstation**, run the **lab processes-kill start** command. The command runs a start script that determines whether the host, **servera**, is reachable on the network.

```
[student@workstation ~]$ lab processes-kill start
```

- 1. On **workstation**, open two terminal windows side by side. In this section, these terminals are referred to as *left* and *right*. In each terminal, use the **ssh** command to log in to **servera** as the **student** user.

```
[student@workstation ~]$ ssh student@servera  
...output omitted...  
[student@servera ~]$
```

- 2. In the *left* window, create a new directory called **/home/student/bin**. In the new directory, create a shell script called **killing**. Make the script executable.

- 2.1. Use the **mkdir** command to create a new directory called **/home/student/bin**.

```
[student@servera ~]$ mkdir /home/student/bin
```

- 2.2. Use the **vim** command to create a script called **killing** in the **/home/student/bin** directory. Press the **i** key to enter Vim interactive mode. Use the **:wq** command to save the file.

```
[student@servera ~]$ vim /home/student/bin/killing  
#!/bin/bash  
while true; do  
    echo -n "$@" >> ~/killing_outfile  
    sleep 5  
done
```

**Note**

The **killing** script runs until terminated. It appends command line arguments to the **~/killing_outfile** once every 5 seconds.

- 2.3. Use the **chmod** command to make the **killing** file executable.

```
[student@servera ~]$ chmod +x /home/student/bin/killing
```

- 3. In the left terminal shell, use the **cd** command to change into the **/home/student/bin/** directory. Start three **killing** processes with the arguments **network**, **interface**, and **connection**, respectively. Start three processes called **network**, **interface**, and **connection**. Use the ampersand (&) to start the processes in the background.

```
[student@servera ~]$ cd /home/student/bin
[student@servera bin]$ killing network &
[1] 3460
[student@servera bin]$ killing interface &
[2] 3482
[student@servera bin]$ killing connection &
[3] 3516
```

Your processes will have different PID numbers.

- 4. In the right terminal shell, use the **tail** command with the **-f** option to confirm that all three processes are appending to the **/home/student/killing_outfile** file.

```
[student@servera ~]$ tail -f ~/killing_outfile
network interface network connection interface network connection interface
  network
...output omitted...
```

- 5. In the left terminal shell, use the **jobs** command to list jobs.

```
[student@servera bin]$ jobs
[1]  Running           killing network &
[2]- Running           killing interface &
[3]+ Running           killing connection &
```

- 6. Use signals to suspend the **network** process. Confirm that the **network** process is **Stopped**. In the right terminal shell, confirm that the **network** process is no longer appending output to the **~/killing_output**.

- 6.1. Use the **kill** with the **-SIGSTOP** option to stop the **network** process. Run the **jobs** to confirm it is stopped.

```
[student@servera bin]$ kill -SIGSTOP %1
[1]+  Stopped                  killing network
[student@servera bin]$ jobs
[1]+  Stopped                  killing network
[2]   Running                  killing interface &
[3]-  Running                  killing connection &
```

- 6.2. In the right terminal shell, look at the output from the **tail** command. Confirm that the word **network** is no longer being appended to the **~/killing_outfile** file.

```
...output omitted...
interface connection interface connection interface connection interface
```

- ▶ 7. In the left terminal shell, terminate the **interface** process using signals. Confirm that the **interface** process has disappeared. In the right terminal shell, confirm that **interface** process output is no longer appended to the **~/killing_outfile** file.

- 7.1. Use the **kill** command with the **-SIGTERM** option to terminate the **interface** process. Run the **jobs** command to confirm that it has been terminated.

```
[student@servera bin]$ kill -SIGTERM %2
[student@servera bin]$ jobs
[1]+  Stopped                  killing network
[2]  Terminated                killing interface
[3]-  Running                  killing connection &
```

- 7.2. In the right terminal shell, look at the output from the **tail** command. Confirm that the word **interface** is no longer being appended to the **~/killing_outfile** file.

```
...output omitted...
connection connection connection connection connection connection connection
connection
```

- ▶ 8. In the left terminal shell, resume the **network** process using signals. Confirm that the **network** process is **Running**. In the right window, confirm that **network** process output is being appended to the **~/killing_outfile** file.

- 8.1. Use the **kill** command with the **-SIGCONT** to resume the **network** process. Run the **jobs** command to confirm that the process is **Running**.

```
[student@servera bin]$ kill -SIGCONT %1
[student@servera bin]$ jobs
[1]+  Running                  killing network &
[3]-  Running                  killing connection &
```

- 8.2. In the right terminal shell, look at the output from the **tail** command. Confirm that the word **network** is being appended to the **~/killing_outfile** file.

```
...output omitted...
network connection network connection network connection network connection
network connection
```

- ▶ 9. In the left terminal shell, terminate the remaining two jobs. Confirm that no jobs remain and that output has stopped.

- 9.1. Use the **kill** command with the **-SIGTERM** option to terminate the **network** process. Use the same command to terminate the **connection** process.

```
[student@servera bin]$ kill -SIGTERM %1
[student@servera bin]$ kill -SIGTERM %3
[1]+  Terminated          killing network
[student@servera bin]$ jobs
[3]+  Terminated          killing connection
```

- ▶ 10. In the left terminal shell, list **tail** processes running in all open terminal shells. Terminate running tail commands. Confirm that the process is no longer running.

- 10.1. Use the **ps** command with the **-ef** option to list all running tail processes. Refine the search using the **grep** command.

```
[student@servera bin]$ ps -ef | grep tail
student  4581 31358  0 10:02 pts/0    00:00:00 tail -f killing_outfile
student  4869  2252  0 10:33 pts/1    00:00:00 grep --color=auto tail
```

- 10.2. Use the **pkill** command with the **-SIGTERM** option to kill the **tail** process. Use the **ps** to confirm it is no longer present.

```
[student@servera bin]$ pkill -SIGTERM tail
[student@servera bin]$ ps -ef | grep tail
student  4874  2252  0 10:36 pts/1    00:00:00 grep --color=auto tail
```

- 10.3. In the right terminal shell, confirm that the **tail** command is no longer running.

```
...output omitted...
network connection network connection network connection Terminated
[student@servera ~]$
```

- ▶ 11. Exit from both terminal windows. Failure to exit from all sessions causes the finish script to fail.

```
[student@servera bin]$ exit
logout
Connection to servera closed.
[student@workstation ~]$
```

```
[student@servera ~]$ exit  
logout  
Connection to servera closed.  
[student@workstation ~]$
```

Finish

On **workstation**, run the **lab processes-kill finish** script to complete this exercise.

```
[student@workstation ~]$ lab processes-kill finish
```

This concludes the guided exercise.

Monitoring Process Activity

Objectives

After completing this section, you should be able to describe what load average is and determine processes responsible for high resource use on a server.

Describing Load Average

Load average is a measurement provided by the Linux kernel that is a simple way to represent the perceived system load over time. It can be used as a rough gauge of how many system resource requests are pending, and to determine whether system load is increasing or decreasing over time.

Every five seconds, the kernel collects the current *load number*, based on the number of processes in runnable and uninterruptible states. This number is accumulated and reported as an exponential moving average over the most recent 1, 5, and 15 minutes.

Understanding the Linux Load Average Calculation

The load average represents the perceived system load over a time period. Linux determines this by reporting how many processes are ready to run on a CPU, and how many processes are waiting for disk or network I/O to complete.

- The load number is a running average of the number of processes that are ready to run (in process state **R**) or are waiting for I/O to complete (in process state **D**).
- Some UNIX systems only consider CPU utilization or run queue length to indicate system load. Linux also includes disk or network utilization because that can have as significant an impact on system performance as CPU load. When experiencing high load averages with minimal CPU activity, examine disk and network activity.

Load average is a rough measurement of how many processes are currently waiting for a request to complete before they can do anything else. The request might be for CPU time to run the process. Alternatively, the request might be for a critical disk I/O operation to complete, and the process cannot be run on the CPU until the request completes, even if the CPU is idle. Either way, system load is impacted and the system appears to run more slowly because processes are waiting to run.

Interpreting Displayed Load Average Values

The **uptime** command is one way to display the current load average. It prints the current time, how long the machine has been up, how many user sessions are running, and the current load average.

```
[user@host ~]$ uptime  
15:29:03 up 14 min,  2 users,  load average: 2.92, 4.48, 5.20
```

The three values for the load average represent the load over the last 1, 5, and 15 minutes. A quick glance indicates whether system load appears to be increasing or decreasing.

If the main contribution to load average is from processes waiting for the CPU, you can calculate the approximate *per CPU* load value to determine whether the system is experiencing significant waiting.

The **lscpu** command can help you determine how many CPUs a system has.

In the following example, the system is a dual-core single socket system with two hyperthreads per core. Roughly speaking, Linux will treat this as a four CPU system for scheduling purposes.

```
[user@host ~]$ lscpu
Architecture:          x86_64
CPU op-mode(s):        32-bit, 64-bit
Byte Order:            Little Endian
CPU(s):                4
On-line CPU(s) list:  0-3
Thread(s) per core:   2
Core(s) per socket:   2
Socket(s):             1
NUMA node(s):          1
...output omitted...
```

For a moment, imagine that the only contribution to the load number is from processes that need CPU time. Then you can 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 processing delay.

```
# From lscpu, the 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 process waiting for a CPU adds a count of 1 to the load number. If one process is running on a CPU, the load number is one, the resource (the CPU) is in use, but there are no requests waiting. If that process is running for a full minute, its contribution to the one-minute load average will be 1.

However, processes uninterruptibly sleeping for critical I/O due to a busy disk or network resource are also included in the count and increase the load average. While not an indication of CPU utilization, these processes are added to the queue count because they are waiting for resources and cannot run on a CPU until they get them. This is still system load due to resource limitations that is causing processes not to run.

Until resource saturation, a load average remains below 1, since tasks are seldom found waiting in queue. Load average only increases when resource saturation causes requests to remain queued and are counted by the load calculation routine. When resource utilization approaches 100%, each additional request starts experiencing service wait time.

A number of additional tools report load average, including **w** and **top**.

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** = Uninterruptible 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 Running processes; default is bold only.
Shift+b	Enables use of bold in display, in the header, and for <i>Running</i> processes.
Shift+h	Toggle threads; show process summary or individual threads.
u , Shift+u	Filter for any user name (effective, real).
Shift+m	Sorts process listing by memory usage, in descending order.
Shift+p	Sorts process listing by processor utilization, in descending order.

Key	Purpose
k ⁽¹⁾	Kill a process. When prompted, enter PID , then signal .
r ⁽¹⁾	Renice a process. When prompted, enter PID , then nice_value .
Shift+w	Write (save) the current display configuration for use at the next top restart.
q	Quit.
f	Manage the columns by enabling or disabling fields. Also allows you to set the sort field for top .
Note:	⁽¹⁾ Not available if top started in secure mode. See top(1) .



References

ps(1), **top(1)**, **uptime(1)**, and **w(1)** man pages

► Guided Exercise

Monitoring Process Activity

In this exercise, you will use the **top** command to dynamically examine running processes and control them.

Outcomes

You should be able to manage processes in real time.

Before You Begin

Log in to **workstation** as **student** using **student** as the password.

On **workstation**, run the **lab processes-monitor start** command. The command runs a start script that determines if the host, **servera**, is reachable on the network.

```
[student@workstation ~]$ lab processes-monitor start
```

- 1. On **workstation** open two terminal windows side by side. These terminals are referred to as *left* and *right*. On each terminal, use the **ssh** command to log in to **servera** as the **student** user.

```
[student@workstation ~]$ ssh student@servera
...output omitted...
[student@servera ~]$
```

- 2. In the *left* terminal shell, create a new directory called **/home/student/bin**. In the new directory create a shell script called **monitor**, which generates artificial CPU load. Ensure the script is executable.
- 2.1. Use the **mkdir** command to create a new directory called **/home/student/bin**.

```
[student@servera ~]$ mkdir /home/student/bin
```

- 2.2. Create a script named **monitor** in the **/home/student/bin** directory with the following content:

```
#!/bin/bash
while true; do
    var=1
    while [[ var -lt 60000 ]]; do
        var=$((var+1))
    done
    sleep 1
done
```

The **monitor** script runs until terminated. It generates artificial CPU load by performing sixty thousand addition problems. It then sleeps for one second, resets the variable, and repeats.

- 2.3. Use the **chmod** command to make the **monitor** file executable.

```
[student@servera ~]$ chmod a+x /home/student/bin/monitor
```

- 3. In the right terminal shell, run the **top** utility. Size the window to be as tall as possible.

```
[student@servera ~]$ top
top - 12:13:03 up 11 days, 58 min, 3 users, load average: 0.00, 0.00, 0.00
Tasks: 113 total, 2 running, 111 sleeping, 0 stopped, 0 zombie
%Cpu(s): 0.2 us, 0.0 sy, 0.0 ni, 99.8 id, 0.0 wa, 0.0 hi, 0.0 si, 0.0 st
MiB Mem : 1829.4 total, 1377.3 free, 193.9 used, 258.2 buff/cache
MiB Swap: 1024.0 total, 1024.0 free, 0.0 used. 1476.1 avail Mem

PID USER      PR  NI    VIRT    RES    SHR S %CPU %MEM     TIME+ COMMAND
5861 root      20   0        0      0      0 I  0.3  0.0  0:00.71 kworker/1:3-
events
6068 student   20   0  273564  4300  3688 R  0.3  0.2  0:00.01 top
  1 root      20   0  178680 13424  8924 S  0.0  0.7  0:04.03 systemd
  2 root      20   0        0      0      0 S  0.0  0.0  0:00.03 kthreadd
  3 root      0 -20        0      0      0 I  0.0  0.0  0:00.00 rcu_gp
...output omitted...
```

- 4. In the left terminal shell use the **lscpu** command to determine the number of logical CPUs on this virtual machine.

```
[student@servera ~]$ lscpu
Architecture:           x86_64
CPU op-mode(s):         32-bit, 64-bit
Byte Order:             Little Endian
CPU(s):                 2
...output omitted...
```

- 5. In the left terminal shell, run a single instance of the **monitor** executable. Use the ampersand (&) to run the process in the background.

```
[student@servera ~]$ monitor &
[1] 6071
```

- 6. In the right terminal shell, 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.
- 7. Note the process ID (PID) for **monitor**. View the CPU percentage for the process, which is expected to hover around 15% to 25%.

```
[student@servera ~]$ top
PID USER      PR  NI    VIRT      RES      SHR S %CPU %MEM     TIME+ COMMAND
071 student    20   0 222448  2964  2716 S 18.7  0.2   0:27.35 monitor
...output omitted...
```

View the load averages. 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.

```
top - 12:23:45 up 11 days,  1:09,  3 users,  load average: 0.21, 0.14, 0.05
```

- ▶ 8. In the left terminal shell, run a second instance of **monitor**. Use the ampersand (&) to run the process in the background.

```
[student@servera ~]$ monitor &
[2] 6498
```

- ▶ 9. In the right terminal shell, note the process ID (PID) for the second **monitor** process. View the CPU percentage for the process, also expected to hover between 15% and 25%.

```
[student@servera ~]$ top
PID USER      PR  NI    VIRT      RES      SHR S %CPU %MEM     TIME+ COMMAND
6071 student    20   0 222448  2964  2716 S 19.0  0.2   1:36.53 monitor
6498 student    20   0 222448  2996  2748 R 15.7  0.2   0:16.34 monitor
...output omitted...
```

View the one minute load average again, which is still less than 1. It is important to wait for at least one minute to allow the calculation to adjust to the new workload.

```
top - 12:27:39 up 11 days,  1:13,  3 users,  load average: 0.36, 0.25, 0.11
```

- ▶ 10. In the left terminal shell, run a third instance of **monitor**. Use the ampersand (&) to run the process in the background.

```
[student@servera ~]$ monitor &
[3] 6881
```

- ▶ 11. In the right terminal shell, note the process ID (PID) for the third **monitor** process. View the CPU percentage for the process, again expected to hover between 15% and 25%.

```
[student@servera ~]$ top
PID USER      PR  NI    VIRT      RES      SHR S %CPU %MEM     TIME+ COMMAND
6881 student    20   0 222448  3032  2784 S 18.6  0.2   0:11.48 monitor
6498 student    20   0 222448  2996  2748 S 15.6  0.2   0:47.86 monitor
6071 student    20   0 222448  2964  2716 S 18.1  0.2   2:07.86 monitor
```

To push the load average above 1, you must start more **monitor** processes. The classroom setup has 2 CPUs so only 3 processes are not enough to stress it. Start three more **monitor** processes. View the one minute load average again, which now is expected to be

above 1. It is important to wait for at least one minute to allow the calculation to adjust to the new workload.

```
[student@servera ~]$ monitor &
[4] 10708
[student@servera ~]$ monitor &
[5] 11122
[student@servera ~]$ monitor &
[6] 11338
```

```
top - 12:42:32 up 11 days, 1:28, 3 users, load average: 1.23, 2.50, 1.54
```

- ▶ 12. When finished observing the load average values, terminate each of the **monitor** processes from within **top**.
 - 12.1. In the right terminal shell, press **k**. Observe the prompt below the headers and above the columns.

```
...output omitted...
PID to signal/kill [default pid = 11338]
```

- 12.2. The prompt has chosen the **monitor** processes at the top of the list. Press **Enter** to kill the process.

```
...output omitted...
Send pid 11338 signal [15/sigterm]
```

- 12.3. Press **Enter** again to confirm the 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.

PID	User	Time	Process	State	CPU %	Memory %	Time	Process			
6498	student	20	0	222448	2996	2748	R	22.9	0.2	5:31.47	monitor
6881	student	20	0	222448	3032	2784	R	21.3	0.2	4:54.47	monitor
11122	student	20	0	222448	2984	2736	R	15.3	0.2	2:32.48	monitor
6071	student	20	0	222448	2964	2716	S	15.0	0.2	6:50.90	monitor
10708	student	20	0	222448	3032	2784	S	14.6	0.2	2:53.46	monitor

- ▶ 13. Repeat the previous step for each remaining **monitor** instance. Confirm that no **monitor** processes remain in **top**.
- ▶ 14. In the right terminal shell, press **q** to exit **top**. Exit from **servera** on both terminal windows.

```
[student@servera ~]$ exit
logout
Connection to servera closed.
[student@workstation ~]$
```

```
[student@servera ~]$ exit  
logout  
Connection to servera closed.  
[student@workstation ~]$
```

Finish

On **workstation**, run the **lab processes-monitor finish** script to complete this exercise.

```
[student@workstation ~]$ lab processes-monitor finish
```

This concludes the guided exercise.

▶ Lab

Monitoring and Managing Linux Processes

Performance Checklist

In this lab, you will locate and manage processes that are using the most resources on a system.

Outcomes

You should be able to manage processes using **top** as a process management tool.

Before You Begin

Log in to **workstation** as **student** using **student** as the password.

On **workstation**, run the **lab processes-review start** command. The command runs a start script to determine whether the host, **serverb**, is reachable on the network.

```
[student@workstation ~]$ lab processes-review start
```

1. On **workstation**, open two terminal windows side by side. In this section, these terminals are referred to as *left* and *right*. On each terminal window, log in to **serverb** as the **student** user.
Create a script called **process101**, which will generate artificial CPU load. Create the script in the **/home/student/bin** directory.

```
#!/bin/bash
while true; do
    var=1
    while [[ var -lt 50000 ]]; do
        var=$((var+1))
    done
    sleep 1
done
```

2. In the right window, run the **top** utility.
3. In the left terminal shell, determine the number of logical CPUs on the virtual machine. Run the **process101** script in the background.
4. In the right terminal shell, observe the **top** display. Toggle between load, threads and memory. Note the process ID (PID) for **process101**. View the CPU percentage. It should hover around 10% to 15%. Ensure that **top** is showing CPU usage once you have viewed load, threads, and memory.
5. Turn off the use of bold in the display. Save this configuration for reuse when top is restarted. Confirm that the changes are saved.

6. Copy the **process101** script to a new file called **process102**. Edit the script to create more artificial CPU load. Increase the load from fifty thousand to one hundred thousand. Start the **process102** process in the background.
7. In the right terminal shell, confirm that the process is running and using the most CPU resources. The load should be hovering between 25% and 35%.
8. The load average is still below 1. Copy **process101** to a new script called **process103**. Increase the addition count to eight hundred thousand. Start **process103** in the background. Confirm that the load average is above 1. It may take a few minutes for the load average to change.
9. In the left terminal shell, become **root**. Suspend the **process101** process. List the remaining jobs. Observe that the process state for **process101** is now **T**.
10. Resume the **process101** process.
11. Terminate **process101**, **process102**, and **process103** using the command line. Confirm that the processes are no longer displayed in **top**.
12. In the left terminal shell, exit from the **root** user. In the right terminal shell stop the **top** command. Exit from **serverb** in both windows.

Evaluation

On workstation, run the **lab processes-review grade** script to confirm success on this exercise.

```
[student@workstation ~]$ lab processes-review grade
```

Finish

On workstation, run the **lab processes-review finish** script to complete the lab.

```
[student@workstation ~]$ lab processes-review finish
```

This concludes the lab.

► Solution

Monitoring and Managing Linux Processes

Performance Checklist

In this lab, you will locate and manage processes that are using the most resources on a system.

Outcomes

You should be able to manage processes using **top** as a process management tool.

Before You Begin

Log in to **workstation** as **student** using **student** as the password.

On **workstation**, run the **lab processes-review start** command. The command runs a start script to determine whether the host, **serverb**, is reachable on the network.

```
[student@workstation ~]$ lab processes-review start
```

1. On **workstation**, open two terminal windows side by side. In this section, these terminals are referred to as *left* and *right*. On each terminal window, log in to **serverb** as the **student** user.
Create a script called **process101**, which will generate artificial CPU load. Create the script in the **/home/student/bin** directory.

```
#!/bin/bash
while true; do
    var=1
    while [[ var -lt 50000 ]]; do
        var=$((var+1))
    done
    sleep 1
done
```

- 1.1. On **workstation**, open two terminal windows side by side. In each terminal, use the **ssh** command to log in to **serverb** as the **student** user.

```
[student@workstation ~]$ ssh student@serverb
...output omitted...
[student@serverb ~]$
```

- 1.2. In the left terminal shell, use the **mkdir** command to create the **/home/student/bin** directory.

```
[student@serverb ~]$ mkdir /home/student/bin
```

- 1.3. In the left terminal shell, use the **vim** command to create the **process101** script. Press the **i** key to enter interactive mode. Type **:wq** to save the file.

```
[student@serverb ~]$ vim /home/student/bin/process101
#!/bin/bash
while true; do
    var=1
    while [[ var -lt 50000 ]]; do
        var=$((var+1))
    done
    sleep 1
done
```

- 1.4. Use the **chmod** command to make the **process101** file executable.

```
[student@serverb ~]$ chmod +x /home/student/bin/process101
```

2. In the right window, run the **top** utility.

- 2.1. In the right window, run the **top** utility. Size the window to be as tall as possible.

```
[student@serverb ~]$ top
top - 13:47:06 up 19 min,  2 users,  load average: 0.00, 0.00, 0.00
Tasks: 110 total,   1 running, 109 sleeping,   0 stopped,   0 zombie
%Cpu(s):  0.0 us,  3.1 sy,  0.0 ni, 96.9 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
MiB Mem : 1829.4 total,   1439.1 free,    171.9 used,   218.4 buff/cache
MiB Swap: 1024.0 total,   1024.0 free,      0.0 used. 1499.6 avail Mem

PID USER      PR  NI      VIRT      RES      SHR S %CPU %MEM     TIME     COMMAND
 1 root      20   0  178536  13488  8996 S  0.0  0.7  0:01.15  systemd
 2 root      20   0          0          0      0 S  0.0  0.0  0:00.00  kthreadd
 3 root      0 -20          0          0      0 I  0.0  0.0  0:00.00  rcu_gp
 4 root      0 -20          0          0      0 I  0.0  0.0  0:00.00  rcu_par_gp
 6 root      0 -20          0          0      0 I  0.0  0.0  0:00.00  kworker/0:0H-
kblockd
...output omitted...
```

3. In the left terminal shell, determine the number of logical CPUs on the virtual machine. Run the **process101** script in the background.

- 3.1. Use the **grep** command to determine the number of logical CPUs.

```
[student@serverb ~]$ grep "model name" /proc/cpuinfo | wc -l
2
```

- 3.2. Use the **cd** command to change into the **/home/student/bin** directory. Run the **process101** script in the background.

```
[student@serverb ~]$ cd /home/student/bin
[student@serverb bin]$ process101 &
[1] 1180
```

4. In the right terminal shell, observe the **top** display. Toggle between load, threads and memory. Note the process ID (PID) for **process101**. View the CPU percentage. It should hover around 10% to 15%. Ensure that **top** is showing CPU usage once you have viewed load, threads, and memory.

4.1. Press **Shift+m**.

```
top - 13:56:24 up 28 min, 2 users, load average: 0.21, 0.08, 0.02
Tasks: 112 total, 2 running, 110 sleeping, 0 stopped, 0 zombie
%Cpu(s): 5.8 us, 1.3 sy, 0.0 ni, 92.8 id, 0.0 wa, 0.0 hi, 0.0 si, 0.0 st
MiB Mem : 1829.4 total, 1438.1 free, 172.7 used, 218.6 buff/cache
MiB Swap: 1024.0 total, 1024.0 free, 0.0 used. 1498.7 avail Mem

PID USER      PR  NI    VIRT    RES    SHR S %CPU %MEM     TIME+ COMMAND
705 root      20   0 409956  34880  33620 S  0.0  1.9  0:00.04 sssd_nss
706 root      20   0 454304  34472  14304 S  0.0  1.8  0:00.62 firewalld
725 root      20   0 611348  28244  14076 S  0.0  1.5  0:00.27 tuned
663 polkitd   20   0 1907312  23876  16040 S  0.0  1.3  0:00.04 polkitd
718 root      20   0 600316  17176  14832 S  0.0  0.9  0:00.06 NetworkManager
...output omitted...
```



Note

Note that when top is switched into *memory* mode, **process101** is no longer the first process. You can press **Shift+p** to return to CPU usage.

4.2. Press **m**.

```
top - 09:32:52 up 20:05, 2 users, load average: 0.18, 0.10, 0.03
Tasks: 112 total, 2 running, 110 sleeping, 0 stopped, 0 zombie
%Cpu(s): 7.8/1.5      9[|||||||]                                ]
MiB Mem : 18.3/1829.4 [|||||||||||||||||]                      ]
MiB Swap: 0.0/1024.0 [                                     ]      ]
PID USER      PR  NI    VIRT    RES    SHR S %CPU %MEM     TIME+ COMMAND
705 root      20   0 409956  34880  33620 S  0.0  1.9  0:00.04 sssd_nss
706 root      20   0 454304  34472  14304 S  0.0  1.8  0:00.62 firewalld
725 root      20   0 611348  28244  14076 S  0.0  1.5  0:00.30 tuned
663 polkitd   20   0 1907312  23876  16040 S  0.0  1.3  0:00.04 polkitd
718 root      20   0 600316  17176  14832 S  0.0  0.9  0:00.07 NetworkManager
...output omitted...
```

4.3. Press **t**.

```
Tasks: 113 total, 2 running, 111 sleeping, 0 stopped, 0 zombie
%Cpu(s): 7.8/1.5      9[|||||||]                                ]
MiB Mem : 1829.4 total, 1436.7 free, 173.7 used, 219.0 buff/cache
MiB Swap: 1024.0 total, 1024.0 free, 0.0 used. 1497.7 avail Mem
```

```

PID USER      PR  NI      VIRT      RES      SHR S %CPU %MEM     TIME+ COMMAND
1180 student   20   0  222448   3056   2808 S 12.0  0.2  1:59.94 process101
 705 root      20   0  409956  34880  33620 S  0.0  1.9  0:00.04 sssd_nss
 706 root      20   0  454304  34472  14304 S  0.0  1.8  0:00.62 firewalld
 725 root      20   0  611348  28244  14076 S  0.0  1.5  0:00.30 tuned
 663 polkitd   20   0 1907312  23876  16040 S  0.0  1.3  0:00.04 polkitd
 718 root      20   0  600316  17176  14832 S  0.0  0.9  0:00.07 NetworkManager
...output omitted...

```

4.4. Press **Shift+p**.

```

top - 09:35:48 up 20:08, 2 users, load average: 0.10, 0.10, 0.04
Tasks: 110 total, 4 running, 106 sleeping, 0 stopped, 0 zombie
%Cpu(s): 6.8 us, 1.0 sy, 0.0 ni, 92.2 id, 0.0 wa, 0.0 hi, 0.0 si, 0.0 st
MiB Mem : 18.3/1829.4 [||||||||||||||||||]
MiB Swap: 0.0/1024.0 []

PID USER      PR  NI      VIRT      RES      SHR S %CPU %MEM     TIME+ COMMAND
27179 student   20   0  222448   3060   2812 R 15.6  0.2  1:13.47 process101
...output omitted...

```

5. Turn off the use of bold in the display. Save this configuration for reuse when top is restarted. Confirm that the changes are saved.

5.1. Press **Shift+b** to switch the use of bold off.

```

top - 19:40:30 up 6:12, 2 users, load average: 0.11, 0.12, 0.09
Tasks: 112 total, 1 running, 111 sleeping, 0 stopped, 0 zombie
%Cpu(s): 7.6/1.5 9[|||||||||]
MiB Mem : 18.2/1829.4 [||||||||||||||||||]
MiB Swap: 0.0/1024.0 []

```

5.2. Press **Shift+w** to save this configuration. The default configuration is stored in **toprc** beneath the **/home/student/.config/procps** directory. In the left terminal shell, confirm that the **toprc** file exists.

```
[student@serverb bin]$ ls -l /home/student/.config/procps/toprc
-rw-rw-r--. 1 student student 966 Feb 18 19:45 /home/student/.config/procps/toprc
```

5.3. In the right terminal shell, exit **top**, and then restart it. Confirm that the new display uses the saved configuration.

```

top - 00:58:21 up 43 min, 2 users, load average: 0.29, 0.28, 0.20
Tasks: 105 total, 1 running, 104 sleeping, 0 stopped, 0 zombie
%Cpu(s): 11.0/1.8 13[|||||||||||||]
MiB Mem : 18.7/1829.0 [||||||||||||||||||]
MiB Swap: 0.0/0.0 []

```

6. Copy the **process101** script to a new file called **process102**. Edit the script to create more artificial CPU load. Increase the load from fifty thousand to one hundred thousand. Start the **process102** process in the background.

- 6.1. In the left terminal shell, use the **cp** command to copy **process101** to **process102**.

```
[student@serverb bin]$ cp process101 process102
```

- 6.2. Use the **vim** command to edit the **process102** script. Increase the addition problems from fifty thousand to one hundred thousand. Enter interactive mode by using the **i**. Type **:wq** to save the file.

```
[student@serverb bin]$ vim process102
#!/bin/bash
while true; do
    var=1
    while [[ var -lt 100000 ]]; do
        var=$((var+1))
    done
    sleep 1
done
```

- 6.3. Start the **process102** process in the background.

```
[student@serverb bin]$ process102 &
[2] 20723
```

- 6.4. Use the **jobs** command to confirm that both processes are running in the background.

```
[student@serverb bin]$ jobs
[1]-  Running                  process101 &
[2]+  Running                  process102 &
```

7. In the right terminal shell, confirm that the process is running and using the most CPU resources. The load should be hovering between 25% and 35%.

- 7.1. In the right terminal shell, confirm that the process is running and using the most CPU resources. The load should be hovering between 25% and 35%.

```
top - 20:14:16 up  6:46,  2 users,  load average: 0.58, 0.34, 0.18
Tasks: 112 total,   2 running, 110 sleeping,   0 stopped,   0 zombie
499 %Cpu(s):  0.0 us,  0.0 sy,  0.0 ni,100.0 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0
      st
500 MiB Mem : 1829.4 total,   1428.7 free,     179.0 used,   221.8 buff/cache
501 MiB Swap: 1024.0 total,   1024.0 free,      0.0 used. 1492.1 avail Mem

      PID USER      PR  NI      VIRT      RES      SHR S %CPU %MEM      TIME+ COMMAND
20723 student    20   0  222448   3016   2764 S 24.7  0.2  0:53.28 process102
1180 student    20   0  222448   3056   2808 S 12.0  0.2 58:01.56 process101
...output omitted...
```



Note

If you do not see **process101** and **process102** at the top of the process list, press **Shift+p** to ensure that top is sorted by CPU utilization.

8. The load average is still below 1. Copy **process101** to a new script called **process103**. Increase the addition count to eight hundred thousand. Start **process103** in the background. Confirm that the load average is above 1. It may take a few minutes for the load average to change.

8.1. In the right terminal shell, confirm that the load average is below 1.

```
top - 20:24:13 up 6:56, 2 users, load average: 0.43, 0.41, 0.29
...output omitted...
```

8.2. In the left terminal shell, use the **cp** command to copy **process101** to a new script called **process103**.

```
[student@serverb bin]$ cp process101 process103
```

8.3. In the left terminal shell, use the **vim** command to edit the **process103** script. Increase the addition count to eight hundred thousand. Enter interactive mode using the **i** key. Type **:wq** to save the file.

```
[student@serverb bin]$ vim process103
#!/bin/bash
while true; do
    var=1
    while [[ var -lt 800000 ]]; do
        var=$((var+1))
    done
    sleep 1
done
```

8.4. Start **process103** in the background. The CPU usage hovers between 60% and 85%.

```
[student@serverb bin]$ process103 &
[3] 22751
```

8.5. Confirm that all three jobs are running in the background.

```
[student@serverb bin]$ jobs
[1]  Running                  process101 &
[2]- Running                  process102 &
[3]+ Running                  process103 &
```

8.6. In the right terminal window confirm that the load average is above 1.

```
top - 20:45:34 up 7:17, 2 users, load average: 1.10, 0.90, 0.64
```

9. In the left terminal shell, become **root**. Suspend the **process101** process. List the remaining jobs. Observe that the process state for **process101** is now **T**.

9.1. Use the **su -** command to become **root**. The password is **redhat**.

```
[student@serverb bin]$ su -
Password: redhat
```

- 9.2. Use the **pkill** command with the **-SIGSTOP** option to suspend the **process101** process.

```
[root@serverb ~]# pkill -SIGSTOP process101
```

- 9.3. In the right terminal shell confirm that **process101** is no longer running.

```
top - 20:52:01 up 7:24, 2 users, load average: 1.19, 1.19, 0.89
Tasks: 112 total, 2 running, 110 sleeping, 0 stopped, 0 zombie
499 %Cpu(s): 0.0 us, 0.0 sy, 0.0 ni, 100.0 id, 0.0 wa, 0.0 hi, 0.0 si, 0.0 st
500 MiB Mem : 1829.4 total, 1428.7 free, 179.0 used, 221.8 buff/cache
501 MiB Swap: 1024.0 total, 1024.0 free, 0.0 used. 1492.1 avail Mem

PID USER PR NI VIRT RES SHR S %CPU %MEM TIME+ COMMAND
24043 student 20 0 222448 2992 2744 R 66.1 0.2 6:59.50 process103
20723 student 20 0 222448 3016 2764 R 29.9 0.2 11:04.84 process102
...output omitted...
```

- 9.4. In the left terminal shell run the **ps jT** command to view the remaining jobs.

```
[root@serverb ~]# ps jT
PPID PID PGID SID TTY TPGID STAT UID TIME COMMAND
...output omitted...
27138 1180 1180 27138 pts/0 28558 T 1000 3:06 /bin/bash /home/student/bin/process101
27138 20723 20723 27138 pts/0 28558 R 1000 1:23 /bin/bash /home/student/bin/process102
27138 24043 24043 27138 pts/0 28558 R 1000 2:35 /bin/bash /home/student/bin/process103
...output omitted...
```

Note that **process101** has a status of **T**. This denotes that the process is currently suspended.

10. Resume the **process101** process.

- 10.1. In the left terminal shell use the **pkill** command with the **-SIGCONT** option to resume the **process101** process.

```
[root@serverb ~]# pkill -SIGCONT process101
```

- 10.2. In the right terminal shell confirm that the process is running again.

```
top - 20:57:02 up 7:29, 2 users, load average: 1.14, 1.20, 0.99
Tasks: 112 total, 2 running, 110 sleeping, 0 stopped, 0 zombie
%Cpu(s): 0.0 us, 0.0 sy, 0.0 ni, 100.0 id, 0.0 wa, 0.0 hi, 0.0 si, 0.0 st
MiB Mem : 1829.4 total, 1428.7 free, 179.0 used, 221.8 buff/cache
```

```
MiB Swap: 1024.0 total, 1024.0 free, 0.0 used. 1492.1 avail Mem

 PID USER PR NI VIRT RES SHR S %CPU %MEM TIME+ COMMAND
24043 student 20 0 222448 2992 2744 R 66.8 0.2 10:40.61 process103
20723 student 20 0 222448 3016 2764 S 24.9 0.2 12:25.10 process102
1180 student 20 0 222448 3056 2808 S 17.9 0.2 64:07.99 process101
```

11. Terminate **process101**, **process102**, and **process103** using the command line. Confirm that the processes are no longer displayed in **top**.

- 11.1. In the left terminal shell, use the **pkill** command terminate **process101**, **process102**, and **process103**.

```
[root@serverb ~]# pkill process101
[root@serverb ~]# pkill process102
[root@serverb ~]# pkill process103
```

- 11.2. In the right terminal shell, confirm that the processes no longer appear in **top**.

```
top - 21:05:06 up 7:37, 2 users, load average: 1.26, 1.29, 1.12
Tasks: 112 total, 2 running, 110 sleeping, 0 stopped, 0 zombie
499 %Cpu(s): 0.0 us, 0.0 sy, 0.0 ni, 100.0 id, 0.0 wa, 0.0 hi, 0.0 si, 0.0 st
500 MiB Mem : 1829.4 total, 1428.7 free, 179.0 used, 221.8 buff/cache
501 MiB Swap: 1024.0 total, 1024.0 free, 0.0 used. 1492.1 avail Mem

 PID USER PR NI VIRT RES SHR S %CPU %MEM TIME+ COMMAND
 1 root 20 0 178536 13488 8996 S 0.0 0.7 0:01.21 systemd
 2 root 20 0 0 0 0 S 0.0 0.0 0:00.00 kthreadd
 3 root 0 -20 0 0 0 I 0.0 0.0 0:00.00 rcu_gp
...output omitted...
```

12. In the left terminal shell, exit from the **root** user. In the right terminal shell stop the **top** command. Exit from **serverb** in both windows.

- 12.1. Use the **exit** command to log out from the **root** user.

```
[root@serverb ~]# exit
logout
[1]  Terminated                  process101
[2]  Terminated                  process102
[3]- Terminated                  process103
```

- 12.2. Exit from all terminal windows.

```
[student@serverb bin]$ exit
[student@workstation ~]$
```

- 12.3. In the right terminal shell, press **q** to quit **top**. Use the **exit** command to log out.

```
[student@serverb ~]$ exit  
logout  
Connection to serverb closed.  
[student@workstation ~]$
```

Evaluation

On workstation, run the **lab processes-review grade** script to confirm success on this exercise.

```
[student@workstation ~]$ lab processes-review grade
```

Finish

On workstation, run the **lab processes-review finish** script to complete the lab.

```
[student@workstation ~]$ lab processes-review finish
```

This concludes the lab.

Summary

In this chapter, you learned:

- A process is a running instance of an executable program. Processes are assigned a state, which can be running, sleeping, stopped, or zombie. The **ps** command is used to list processes.
- Each terminal is its own session and can have foreground process and independent background processes. The **jobs** command displays processes within a terminal session.
- A signal is a software interrupt that reports events to an executing program. The **kill**, **pkill**, and **killall** commands use signals to control processes.
- Load average is an estimate of how busy the system is. To display load average values, you can use the **top**, **uptime**, or **w** command.

Chapter 9

Controlling Services and Daemons

Goal

Control and monitor network services and system daemons using Systemd.

Objectives

- List system daemons and network services started by the **systemd** service and socket units.
- Control system daemons and network services, using **systemctl**.

Sections

- Identifying Automatically Started System Processes (and Guided Exercise)
- Controlling System Services (and Guided Exercise)

Lab

Controlling Services and Daemons

Identifying Automatically Started System Processes

Objectives

After completing this section, you should be able to list system daemons and network services started by **systemd** service and socket units.

Introduction to **systemd**

The **systemd** daemon manages startup for Linux, including service startup and service management in general. It activates system resources, server daemons, and other processes both at boot time and on a running system.

Daemons are processes that either 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. It is a convention for names of many daemon programs to end in the letter **d**.

A service in the **systemd** sense 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**).

In Red Hat Enterprise Linux, the first process that starts (PID 1) is **systemd**. A few of the features provided by **systemd** include:

- Parallelization capabilities (starting multiple services simultaneously), 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. For example, a network-dependent service will not attempt to start up until the network is available.
- A method of tracking related processes together by using Linux control groups.

Describing Service Units

systemd uses *units* to manage different types of objects. 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 that **systemd** should monitor. If a client connects to the socket, **systemd** will start a daemon and pass the connection to it. Socket units are used to delay the start of a service at boot time and to start less frequently used services 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.

The **systemctl** command is used to manage units. For example, display available unit types with the **systemctl -t help** command.

**Important**

When using **systemctl**, you can abbreviate unit names, process tree entries, and unit descriptions.

Listing Service Units

You use the **systemctl** command to explore the current state of the system. For example, the following command lists all currently loaded service units, paginating the output using **less**.

```
[root@host ~]# systemctl list-units --type=service
UNIT                      LOAD ACTIVE SUB   DESCRIPTION
atd.service                loaded active running Job spooling tools
auditd.service              loaded active running Security Auditing Service
chronyd.service              loaded active running NTP client/server
crond.service                loaded active running Command Scheduler
dbus.service                 loaded active running D-Bus System Message Bus
...output omitted...
```

The above output limits the type of unit listed to service units with the **--type=service** option. The output has the following columns:

Columns in the **systemctl list-units** Command Output

UNIT

The service unit name.

LOAD

Whether **systemd** properly parsed the unit's configuration and loaded the unit into memory.

ACTIVE

The high-level activation state of the unit. This information indicates whether the unit has started successfully or not.

SUB

The low-level activation state of the unit. This information indicates more detailed information about the unit. The information varies based on unit type, state, and how the unit is executed.

DESCRIPTION

The short description of the unit.

By default, the **systemctl list-units --type=service** command lists only the service units with **active** activation states. The **--all** option lists all service units regardless of the activation states. Use the **--state=** option to filter by the values in the **LOAD**, **ACTIVE**, or **SUB** fields.

```
[root@host ~]# systemctl list-units --type=service --all
UNIT                      LOAD ACTIVE SUB   DESCRIPTION
atd.service                loaded active running Job spooling tools
auditd.service              loaded active running Security Auditing ...
auth-rpcgss-module.service loaded inactive dead    Kernel Module ...
chronyd.service              loaded active running NTP client/server
cpupower.service             loaded inactive dead    Configure CPU power ...
crond.service                loaded active running Command Scheduler
```

```
[root@host ~]# systemctl
● dbus.service           loaded active running D-Bus System Message Bus
● display-manager.service not-found inactive dead     display-manager.service
...output omitted...
```

The **systemctl** command without any arguments lists units that are both loaded and active.

```
[root@host ~]# systemctl
UNIT                      LOAD ACTIVE SUB DESCRIPTION
proc-sys-fs-binfmt_misc.automount    loaded active waiting Arbitrary...
sys-devices-....device            loaded active plugged   Virtio network...
sys-subsystem-net-devices-ens3.device loaded active plugged   Virtio network...
...
-.mount                     loaded active mounted  Root Mount
boot.mount                  loaded active mounted  /boot
...
systemd-ask-password-plymouth.path  loaded active waiting  Forward Password...
systemd-ask-password-wall.path    loaded active waiting  Forward Password...
init.scope                  loaded active running   System and Servi...
session-1.scope             loaded active running   Session 1 of...
atd.service                 loaded active running   Job spooling tools
audited.service             loaded active running   Security Auditing...
chronyd.service             loaded active running   NTP client/server
crond.service               loaded active running   Command Scheduler
...output omitted...
```

The **systemctl list-units** command displays units that the **systemd** service attempts to parse and load into memory; it does not display installed, but not enabled, services. To see the state of all unit files installed, use the **systemctl list-unit-files** command. For example:

```
[root@host ~]# systemctl list-unit-files --type=service
UNIT FILE                         STATE
arp-ethers.service                disabled
atd.service                       enabled
audited.service                   enabled
auth-rpcgss-module.service       static
autovt@.service                  enabled
blk-availability.service         disabled
...output omitted...
```

In the output of the **systemctl list-units-files** command, valid entries for the **STATE** field are **enabled**, **disabled**, **static**, and **masked**.

Viewing Service States

View the status of a specific unit 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@host ~]# systemctl status sshd.service
● sshd.service - OpenSSH server daemon
  Loaded: loaded (/usr/lib/systemd/system/sshd.service; enabled; vendor preset: enabled)
  Active: active (running) since Thu 2019-02-14 12:07:45 IST; 7h ago
    Main PID: 1073 (sshd)
```

```
CGroup: /system.slice/sshd.service
└─1073 /usr/sbin/sshd -D ...

Feb 14 11:51:39 host.example.com systemd[1]: Started OpenSSH server daemon.
Feb 14 11:51:39 host.example.com sshd[1073]: Could not load host key: /et...y
Feb 14 11:51:39 host.example.com sshd[1073]: Server listening on 0.0.0.0 ....
Feb 14 11:51:39 host.example.com sshd[1073]: Server listening on :: port 22.
Feb 14 11:53:21 host.example.com sshd[1270]: error: Could not load host k...y
Feb 14 11:53:22 host.example.com sshd[1270]: Accepted password for root f...2
...output omitted...
```

This command displays the current status of the service. The meaning of the fields are:

Service Unit Information

Field	Description
Loaded	Whether the service unit is loaded into memory.
Active	Whether the service unit is running and if so, how long it has been running.
Main PID	The main process ID of the service, including the command name.
Status	Additional information about the service.

Several keywords indicating the state of the service can be found in the status output:

Service States in the Output of systemctl

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	Is started at boot time.
disabled	Is not set to be started at boot time.
static	Cannot 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 Red Hat Enterprise Linux 6 and earlier.

Verifying the Status of a Service

The **systemctl** command provides methods for verifying the specific states of a service. For example, use the following command to verify that the a service unit is currently active (running):

```
[root@host ~]# systemctl is-active sshd.service
active
```

The command returns state of the service unit, which is usually **active** or **inactive**.

Run the following command to verify whether a service unit is enabled to start automatically during system boot:

```
[root@host ~]# systemctl is-enabled sshd.service
enabled
```

The command returns whether the service unit is enabled to start at boot time, which is usually **enabled** or **disabled**.

To verify whether the unit failed during startup, run the following command:

```
[root@host ~]# systemctl is-failed sshd.service
active
```

The command either returns **active** if it is properly running or **failed** if an error has occurred during startup. In case the unit was stopped, it returns **unknown** or **inactive**.

To list all the failed units, run the **systemctl --failed --type=service** command.

**References**

systemd(1), **systemd.unit(5)**, **systemd.service(5)**, **systemd.socket(5)**, and **systemctl(1)** man pages

For more information, refer to the *Managing services with systemd* chapter in the *Red Hat Enterprise Linux 8 Configuring basic system settings Guide* at https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/8/html-single/configuring_basic_system_settings/managing-services-with-systemd_configuring-basic-system-settings#managing-services-with-systemd_configuring-basic-system-settings

► Guided Exercise

Identifying Automatically Started System Processes

In this exercise, you will list installed service units and identify which services are currently enabled and active on a server.

Outcomes

You should be able to list installed service units and identify active and enabled services on the system.

Before You Begin

Log in as the **student** user on **workstation** using **student** as the password.

From **workstation**, run the **lab services-identify start** command. The command runs a start script that determines if the host, **servera**, is reachable on the network.

```
[student@workstation ~]$ lab services-identify start
```

- 1. Use the **ssh** command to log in to **servera** as the **student** user. The systems are configured to use SSH keys for authentication, therefore a password is not required to log in to **servera**.

```
[student@workstation ~]$ ssh student@servera
...output omitted...
[student@servera ~]$
```

- 2. List all service units installed on **servera**.

```
[student@servera ~]$ systemctl list-units --type=service
UNIT                  LOAD   ACTIVE   SUB      DESCRIPTION
atd.service           loaded  active  running Job spooling tools
auditd.service        loaded  active  running Security Auditing Service
chronyd.service       loaded  active  running NTP client/server
crond.service         loaded  active  running Command Scheduler
dbus.service          loaded  active  running D-Bus System Message Bus
...output omitted...
```

Press **q** to exit the command.

- 3. List all socket units, active and inactive, on **servera**.

```
[student@servera ~]$ systemctl list-units --type=socket --all
UNIT                  LOAD   ACTIVE   SUB      DESCRIPTION
dbus.socket           loaded  active  running D-Bus System Message Bus Socket
```

```

dm-event.socket      loaded active  listening Device-mapper event daemon FIFOs
lvm2-lvmpolld.socket loaded active  listening LVM2 poll daemon socket
...output omitted...
systemd-udevd-control.socket  loaded active  running  udev Control Socket
systemd-udevd-kernel.socket   loaded active  running  udev Kernel Socket

LOAD   = Reflects whether the unit definition was properly loaded.
ACTIVE = The high-level unit activation state, i.e. generalization of SUB.
SUB    = The low-level unit activation state, values depend on unit type.

12 loaded units listed.
To show all installed unit files use 'systemctl list-unit-files'.

```

- ▶ 4. Explore the status of the **chrony** service. This service is used for network time synchronization (NTP).

- 4.1. Display the status of the **chrony** service. Note the process ID of any active daemon.

```

[student@servera ~]$ systemctl status chronyd
● chronyd.service - NTP client/server
  Loaded: loaded (/usr/lib/systemd/system/chronyd.service; enabled; vendor
  preset: enabled)
  Active: active (running) since Wed 2019-02-06 12:46:57 IST; 4h 7min ago
    Docs: man:chronyd(8)
          man:chrony.conf(5)
  Process: 684 ExecStartPost=/usr/libexec/chrony-helper update-daemon
  (code=exited, status=0/SUCCESS)
  Process: 673 ExecStart=/usr/sbin/chronyd $OPTIONS (code=exited, status=0/
  SUCCESS)
  Main PID: 680 (chronyd)
    Tasks: 1 (limit: 11406)
   Memory: 1.5M
      CGroup: /system.slice/chronyd.service
              └─680 /usr/sbin/chronyd

... servera.lab.example.com systemd[1]: Starting NTP client/server...
...output omitted...
... servera.lab.example.com systemd[1]: Started NTP client/server.
... servera.lab.example.com chronyd[680]: Source 172.25.254.254 offline
... servera.lab.example.com chronyd[680]: Source 172.25.254.254 online
... servera.lab.example.com chronyd[680]: Selected source 172.25.254.254

```

Press **q** to exit the command.

- 4.2. Confirm that the listed daemon is running. In the preceding command, the output of the process ID associated with the **chrony** service is 680. The process ID might differ on your system.

```

[student@servera ~]$ ps -p 680
  PID TTY      TIME CMD
 680 ?        00:00:00 chronyd

```

- ▶ 5. Explore the status of the **sshd** service. This service is used for secure encrypted communication between systems.

- 5.1. Determine whether the **sshd** service is enabled to start at system boot.

```
[student@servera ~]$ systemctl is-enabled sshd
enabled
```

- 5.2. Determine if the **sshd** service is active without displaying all of the status information.

```
[student@servera ~]$ systemctl is-active sshd
active
```

- 5.3. Display the status of the **sshd** service.

```
[student@servera ~]$ systemctl status sshd
● sshd.service - OpenSSH server daemon
  Loaded: loaded (/usr/lib/systemd/system/sshd.service; enabled; vendor preset: enabled)
  Active: active (running) since Wed 2019-02-06 12:46:58 IST; 4h 21min ago
    Docs: man:sshd(8)
          man:sshd_config(5)
  Main PID: 720 (sshd)
     Tasks: 1 (limit: 11406)
   Memory: 5.8M
      CGroup: /system.slice/sshd.service
              └─720 /usr/sbin/sshd -D -oCiphers=aes256-gcm@openssh.com,
                chacha20-poly1305@openssh.com,aes256-ctr,
                aes256-cbc,aes128-gcm@openssh.com,aes128-ctr,
                aes128-cbc -oMACs= hmac-sha2-256-etm@openssh.com,hmac-sha>

... servera.lab.example.com systemd[1]: Starting OpenSSH server daemon...
... servera.lab.example.com sshd[720]: Server listening on 0.0.0.0 port 22.
... servera.lab.example.com systemd[1]: Started OpenSSH server daemon.
... servera.lab.example.com sshd[720]: Server listening on :: port 22.
...output omitted...
... servera.lab.example.com sshd[1380]: pam_unix(sshd:session): session opened for user student by (uid=0)
```

Press **q** to exit the command.

- 6. List the enabled or disabled states of all service units.

```
[student@servera ~]$ systemctl list-unit-files --type=service
UNIT FILE                                STATE
arp-ethers.service                         disabled
atd.service                                enabled
auditd.service                            enabled
auth-rpcgss-module.service    static
autovt@.service                           enabled
blk-availability.service      disabled
chrony-dnssrv@.service      static
chrony-wait.service                     disabled
chronyd.service                          enabled
...output omitted...
```

Press **q** to exit the command.

► 7. Exit from **servera**.

```
[student@servera ~]$ exit  
logout  
Connection to servera closed.  
[student@workstation]$
```

Finish

On **workstation**, run the **lab services-identify finish** script to complete this exercise.

```
[student@workstation ~]$ lab services-identify finish
```

This concludes the guided exercise.

Controlling System Services

Objectives

After completing this section, you should be able to control system daemons and network services, using `systemctl`.

Starting and Stopping Services

Services need to be stopped or started manually for a number of reasons: perhaps the service needs to be updated; the configuration file may need to be changed; or a service may need to be uninstalled; or an administrator may manually start an infrequently used service.

To start a service, first verify that it is not running with `systemctl status`. Then, use the `systemctl start` command as the `root` user (using `sudo` if necessary). The example below shows how to start the `sshd.service` service:

```
[root@host ~]# systemctl start sshd.service
```

The `systemd` service looks for `.service` files for service management in commands in the absence of the service type with the service name. Thus the above command can be executed as:

```
[root@host ~]# systemctl start sshd
```

To stop a currently running service, use the `stop` argument with the `systemctl` command. The example below shows how to stop the `sshd.service` service:

```
[root@host ~]# systemctl stop sshd.service
```

Restarting and Reloading Services

During a restart of a running service, the service is stopped and then started. On the restart of service, the process ID changes and a new process ID gets associated during the startup. To restart a running service, use the `restart` argument with the `systemctl` command. The example below shows how to restart the `sshd.service` service:

```
[root@host ~]# systemctl restart sshd.service
```

Some services have the ability to reload their configuration files without requiring a restart. This process is called a service *reload*. Reloading a service does not change the process ID associated with various service processes. To reload a running service, use the `reload` argument with the `systemctl` command. The example below shows how to reload the `sshd.service` service after configuration changes:

```
[root@host ~]# systemctl reload sshd.service
```

In case you are not sure whether the service has the functionality to reload the configuration file changes, use the **reload-or-restart** argument with the **systemctl** command. The command reloads the configuration changes if the reloading functionality is available. Otherwise, the command restarts the service to implements the new configuration changes:

```
[root@host ~]# systemctl reload-or-restart sshd.service
```

Listing Unit Dependencies

Some services require that other services be running first, creating dependencies on the other services. Other services are not started at boot time but rather only on demand. In both cases, systemd and **systemctl** start services as needed whether to resolve the dependency or to start an infrequently used service. For example, if the CUPS print service is not running and a file is placed into the print spool directory, then the system will start CUPS-related daemons or commands to satisfy the print request.

```
[root@host ~]# 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 disables the dependencies.

The **systemctl list-dependencies UNIT** command displays a hierarchy mapping of dependencies to start the service unit. To list reverse dependencies (units that depend on the specified unit), use the **--reverse** option with the command.

```
[root@host ~]# systemctl list-dependencies sshd.service
sshd.service
• └─system.slice
• └─sshd-keygen.target
•   | └─sshd-keygen@ecdsa.service
•   | └─sshd-keygen@ed25519.service
•   | └─sshd-keygen@rsa.service
•   └─sysinit.target
...output omitted...
```

Masking and Unmasking Services

At times, a system may have different services installed that are conflicting with each other. For example, there are multiple methods to manage mail servers (**postfix** and **sendmail**, for example). Masking a service prevents an administrator from accidentally starting a service that conflicts with others. Masking creates a link in the configuration directories to the **/dev/null** file which prevents the service from starting.

```
[root@host ~]# systemctl mask sendmail.service
Created symlink /etc/systemd/system/sendmail.service → /dev/null.
```

```
[root@host ~]# systemctl list-unit-files --type=service
UNIT FILE                                     STATE
...output omitted...
sendmail.service                               masked
...output omitted...
```

Attempting to start a masked service unit fails with the following output:

```
[root@host ~]# systemctl start sendmail.service
Failed to start sendmail.service: Unit sendmail.service is masked.
```

Use the **systemctl unmask** command to unmask the service unit.

```
[root@host ~]# systemctl unmask sendmail
Removed /etc/systemd/system/sendmail.service.
```



Important

A disabled service can be started manually or by other unit files but it does not start automatically at boot. A masked service does not start manually or automatically.

Enabling Services to Start or Stop at Boot

Starting a service on a running system does not guarantee that the service automatically starts when the system reboots. Similarly, stopping a service on a running system does not keep it from starting again when the system reboots. Creating links in the **systemd** configuration directories enables the service to start at boot. The **systemctl** commands creates and removes these links.

To start a service at boot, use the **systemctl enable** command.

```
[root@root ~]# systemctl enable sshd.service
Created symlink /etc/systemd/system/multi-user.target.wants/sshd.service → /usr/
lib/systemd/system/sshd.service.
```

The above command creates a symbolic link from the service unit file, usually in the **/usr/lib/systemd/system** directory, to the location on disk where **systemd** looks for files, which is in the **/etc/systemd/system/TARGETNAME.target.wants** directory. Enabling a service does not start the service in the current session. To start the service and enable it to start automatically during boot, execute both the **systemctl start** and **systemctl enable** commands.

To disable the service from starting automatically, use the following command, which removes the symbolic link created while enabling a service. Note that disabling a service does not stop the service.

```
[root@host ~]# systemctl disable sshd.service
Removed /etc/systemd/system/multi-user.target.wants/sshd.service.
```

To verify whether the service is enabled or disable, use the **systemctl is-enabled** command.

Summary of systemctl Commands

Services can be started and stopped on a running system and enabled or disabled for an automatic start at boot time.

Useful Service Management Commands

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 the 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 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

For more information, refer to the *Managing system services* chapter in the *Red Hat Enterprise Linux 8 Configuring basic system settings Guide* at
https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/8/html-single/configuring_basic_system_settings/managing-services-with-systemd_configuring-basic-system-settings#managing-system-services_managing-services-with-systemd

► Guided Exercise

Controlling System Services

In this exercise, you will use **systemctl** to stop, start, restart, reload, enable, and disable a systemd-managed service.

Outcomes

You should be able to use the **systemctl** command to control **systemd**-managed services.

Before You Begin

Log in as the **student** user on **workstation** using **student** as the password.

From **workstation**, run the **lab services-control start** command. The command runs a start script that determines whether the host, **servera**, is reachable on the network. The script also ensures that the **sshd** and **chronyd** services are running on **servera**.

```
[student@workstation ~]$ lab services-control start
```

- 1. Use the **ssh** command to log in to **servera** as the **student** user. The systems are configured to use SSH keys for authentication, and therefore a password is not required.

```
[student@workstation ~]$ ssh student@servera
...output omitted...
[student@servera ~]$
```

- 2. Execute the **systemctl restart** and **systemctl reload** commands on the **sshd** service. Observe the different results of executing these commands.

- 2.1. Display the status of the **sshd** service. Note the process ID of the **sshd** daemon.

```
[student@servera ~]$ systemctl status sshd
● sshd.service - OpenSSH server daemon
  Loaded: loaded (/usr/lib/systemd/system/sshd.service; enabled; vendor preset: enabled)
  Active: active (running) since Wed 2019-02-06 23:50:42 EST; 9min ago
    Docs: man:sshd(8)
          man:sshd_config(5)
  Main PID: 759 (sshd)
     Tasks: 1 (limit: 11407)
    Memory: 5.9M
  ...output omitted...
```

Press **q** to exit the command.

- 2.2. Restart the **sshd** service and view the status. The process ID of the daemon must change.

```
[student@servera ~]$ sudo systemctl restart sshd
[sudo] password for student: student
[student@servera ~]$ systemctl status sshd
● sshd.service - OpenSSH server daemon
  Loaded: loaded (/usr/lib/systemd/system/sshd.service; enabled; vendor preset: enabled)
  Active: active (running) since Wed 2019-02-06 23:50:42 EST; 9min ago
    Docs: man:sshd(8)
          man:sshd_config(5)
Main PID: 1132 (sshd)
  Tasks: 1 (limit: 11407)
  Memory: 5.9M
...output omitted...
```

In the preceding output, notice that the process ID changed from 759 to 1132 (on your system, the numbers likely will be different). Press **q** to exit the command.

- 2.3. Reload the **sshd** service and view the status. The process ID of the daemon must not change and connections are not interrupted.

```
[student@servera ~]$ sudo systemctl reload sshd
[student@servera ~]$ systemctl status sshd
● sshd.service - OpenSSH server daemon
  Loaded: loaded (/usr/lib/systemd/system/sshd.service; enabled; vendor preset: enabled)
  Active: active (running) since Wed 2019-02-06 23:50:42 EST; 9min ago
    Docs: man:sshd(8)
          man:sshd_config(5)
Main PID: 1132 (sshd)
  Tasks: 1 (limit: 11407)
  Memory: 5.9M
...output omitted...
```

Press **q** to exit the command.

- 3. Verify that the **chronyd** service is running.

```
[student@servera ~]$ systemctl status chronyd
● chronyd.service - NTP client/server
  Loaded: loaded (/usr/lib/systemd/system/chronyd.service; enabled; vendor
  preset: enabled)
  Active: active (running) since Wed 2019-02-06 23:50:38 EST; 1h 25min ago
    Docs: man:chronyd(8)
...output omitted...
```

Press **q** to exit the command.

- 4. Stop the **chronyd** service and view the status.

```
[student@servera ~]$ sudo systemctl stop chronyd
[student@servera ~]$ systemctl status chronyd
● chronyd.service - NTP client/server
  Loaded: loaded (/usr/lib/systemd/system/chronyd.service; enabled; vendor
  preset: enabled)
  Active: inactive (dead)
```

```
Active: inactive (dead) since Thu 2019-02-07 01:20:34 EST; 44s ago
...output omitted...
... servera.lab.example.com chronyd[710]: System clock wrong by 1.349113 seconds,
adjustment started
... servera.lab.example.com systemd[1]: Stopping NTP client/server...
... servera.lab.example.com systemd[1]: Stopped NTP client/server.
```

Press **q** to exit the command.

- 5. Determine if the **chronyd** service is enabled to start at system boot.

```
[student@server ~]$ systemctl is-enabled chronyd
enabled
```

- 6. Reboot **servera**, then view the status of the **chronyd** service.

```
[student@servera ~]$ sudo systemctl reboot
Connection to servera closed by remote host.
Connection to servera closed.
[student@workstation ~]$
```

Log in as the **student** user on **servera** and view the status of the **chronyd** service.

```
[student@workstation ~]$ ssh student@servera
...output omitted...
[student@servera ~]$ systemctl status chronyd
● chronyd.service - NTP client/server
  Loaded: loaded (/usr/lib/systemd/system/chronyd.service; enabled; vendor
  preset: enabled)
    Active: active (running) since Thu 2019-02-07 01:48:26 EST; 5min ago
      ...output omitted...
```

Press **q** to exit the command.

- 7. Disable the **chronyd** service so that it does not start at system boot, then view the status of the service.

```
[student@servera ~]$ sudo systemctl disable chronyd
[sudo] password for student: student
Removed /etc/systemd/system/multi-user.target.wants/chronyd.service.
[student@servera ~]$ systemctl status chronyd
● chronyd.service - NTP client/server
  Loaded: loaded (/usr/lib/systemd/system/chronyd.service; disabled; vendor
  preset: enabled)
    Active: active (running) since Thu 2019-02-07 01:48:26 EST; 5min ago
      ...output omitted...
```

Press **q** to exit the command.

- 8. Reboot **servera**, then view the status of the **chrony** service.

```
[student@servera ~]$ sudo systemctl reboot
Connection to servera closed by remote host.
Connection to servera closed.
[student@workstation ~]$
```

Log in as the **student** user on **servera** and view the status of the **chrony** service.

```
[student@workstation ~]$ ssh student@servera
...output omitted...
[student@servera ~]$ systemctl status chronyd
● chronyd.service - NTP client/server
  Loaded: loaded (/usr/lib/systemd/system/chronyd.service; disabled; vendor
  preset: enabled)
    Active: inactive (dead)
      Docs: man:chronyd(8)
            man:chrony.conf(5)
```

- 9. Exit from **servera**.

```
[student@servera ~]$ exit
logout
Connection to servera closed.
[student@workstation]$
```

Finish

On **workstation**, run the **lab services-control finish** script to complete this exercise.

```
[student@workstation ~]$ lab services-control finish
```

This concludes the guided exercise.

▶ Lab

Controlling Services and Daemons

Performance Checklist

In this lab, you will configure several services to be enabled or disabled, running or stopped, based on a specification that is provided to you.

Outcomes

You should be able to enable, disable, start, and stop services.

Before You Begin

Log in as the **student** user on **workstation** using **student** as the password.

From **workstation**, run the **lab services-review start** command. The command runs a start script that determines whether the host, **serverb**, is reachable on the network. The script also ensures that the **psacct** and **rsyslog** services are configured appropriately on **serverb**.

```
[student@workstation ~]$ lab services-review start
```

1. On **serverb**, start the **psacct** service.
2. Configure the **psacct** service to start at system boot.
3. Stop the **rsyslog** service.
4. Configure the **rsyslog** service so that it does not start at system boot.
5. Reboot **serverb** before evaluating the lab.

Evaluation

On **workstation**, run the **lab services-review grade** script to confirm success on this lab.

```
[student@workstation ~]$ lab services-review grade
```

Finish

On **workstation**, run the **lab services-review finish** script to complete this lab.

```
[student@workstation ~]$ lab services-review finish
```

This concludes the lab.

► Solution

Controlling Services and Daemons

Performance Checklist

In this lab, you will configure several services to be enabled or disabled, running or stopped, based on a specification that is provided to you.

Outcomes

You should be able to enable, disable, start, and stop services.

Before You Begin

Log in as the **student** user on **workstation** using **student** as the password.

From **workstation**, run the **lab services-review start** command. The command runs a start script that determines whether the host, **serverb**, is reachable on the network. The script also ensures that the **psacct** and **rsyslog** services are configured appropriately on **serverb**.

```
[student@workstation ~]$ lab services-review start
```

1. On **serverb**, start the **psacct** service.

- 1.1. Use the **ssh** command to log in to **serverb** as the **student** user.

```
[student@workstation ~]$ ssh student@serverb
...output omitted...
[student@serverb ~]$
```

- 1.2. Use the **systemctl** command to verify the status of the **psacct** service. Notice that **psacct** is stopped and disabled to start at boot time.

```
[student@serverb ~]$ systemctl status psacct
● psacct.service - Kernel process accounting
  Loaded: loaded (/usr/lib/systemd/system/psacct.service; disabled; vendor
  preset: disabled)
    Active: inactive (dead)
```

- 1.3. Start the **psacct** service.

```
[student@serverb ~]$ sudo systemctl start psacct
[sudo] password for student: student
[student@serverb ~]$
```

- 1.4. Verify that the **psacct** service is running.

```
[student@serverb ~]$ systemctl is-active psacct
active
```

2. Configure the **psacct** service to start at system boot.

2.1. Enable the **psacct** service to start at system boot.

```
[student@serverb ~]$ sudo systemctl enable psacct
Created symlink /etc/systemd/system/multi-user.target.wants/psacct.service → /usr/
lib/systemd/system/psacct.service.
```

2.2. Verify that the **psacct** service is enabled to start at system boot.

```
[student@serverb ~]$ systemctl is-enabled psacct
enabled
```

3. Stop the **rsyslog** service.

3.1. Use the **systemctl** command to verify the status of the **rsyslog** service. Notice that the **rsyslog** service is running and enabled to start at boot time.

```
[student@serverb ~]$ systemctl status rsyslog
● rsyslog.service - System Logging Service
  Loaded: loaded (/usr/lib/systemd/system/rsyslog.service; enabled; vendor
  preset: enabled)
  Active: active (running) since Fri 2019-02-08 10:16:00 IST; 2h 34min ago
    ...output omitted...
```

Press **q** to exit the command.

3.2. Stop the **rsyslog** service.

```
[student@serverb ~]$ sudo systemctl stop rsyslog
[sudo] password for student: student
[student@serverb ~]$
```

3.3. Verify that the **rsyslog** service is stopped.

```
[student@serverb ~]$ systemctl is-active rsyslog
inactive
```

4. Configure the **rsyslog** service so that it does not start at system boot.

4.1. Disable the **rsyslog** service so that it does not start at system boot.

```
[student@serverb ~]$ sudo systemctl disable rsyslog
Removed /etc/systemd/system/syslog.service.
Removed /etc/systemd/system/multi-user.target.wants/rsyslog.service.
```

4.2. Verify that the **rsyslog** is disabled to start at system boot.

```
[student@serverb ~]$ systemctl is-enabled rsyslog  
disabled
```

5. Reboot **serverb** before evaluating the lab.

```
[student@serverb ~]$ sudo systemctl reboot  
Connection to serverb closed by remote host.  
Connection to serverb closed.  
[student@workstation ~]$
```

Evaluation

On **workstation**, run the **lab services-review grade** script to confirm success on this lab.

```
[student@workstation ~]$ lab services-review grade
```

Finish

On **workstation**, run the **lab services-review finish** script to complete this lab.

```
[student@workstation ~]$ lab services-review finish
```

This concludes the lab.

Summary

In this chapter, you learned:

- **systemd** provides a method for activating system resources, server daemons, and other processes, both at boot time and on a running system.
- Use the **systemctl** to start, stop, reload, enable, and disable services.
- Use the **systemctl status** command to determine the status of system daemons and network services started by **systemd**.
- The **systemctl list-dependencies** command lists all service units upon which a specific service unit depends.
- **systemd** can mask a service unit so that it does not run even to satisfy dependencies.

Chapter 10

Configuring and Securing SSH

Goal

Configure secure command-line service on remote systems, using OpenSSH.

Objectives

- Log in to a remote system and run commands using `ssh`.
- Configure key-based authentication for a user account to log in to remote systems securely without a password.
- Restrict direct logins as root and disable password-based authentication for the OpenSSH service.

Sections

- Accessing the Remote Command Line with SSH (and Guided Exercise)
- Configuring SSH Key-Based Authentication (and Guided Exercise)
- Customizing OpenSSH Service Configuration (and Guided Exercise)

Lab

Configuring and Securing SSH

Accessing the Remote Command Line with SSH

Objectives

After completing this section, you should be able log into a remote system and run commands using **ssh**.

What is OpenSSH?

OpenSSH implements the Secure Shell or SSH protocol in the Red Hat Enterprise Linux systems. The SSH protocol enables systems to communicate in an encrypted and secure fashion over an insecure network.

You can use the **ssh** command to create a secure connection to a remote system, authenticate as a specific user, and get an interactive shell session on the remote system as that user. You may also use the **ssh** command to run an individual command on the remote system without running an interactive shell.

Secure Shell Examples

The following **ssh** command would log you in on the remote server **remotehost** using the same user name as the current local user. In this example, the remote system prompts you to authenticate with that user's password.

```
[user01@host ~]$ ssh remotehost  
user01@remotehost's password: redhat  
...output omitted...  
[user01@remotehost ~]$
```

You can the **exit** command to log out of the remote system.

```
[user01@remotehost ~]$ exit  
logout  
Connection to remotehost closed.  
[user01@host ~]$
```

The next **ssh** command would log you in on the remote server **remotehost** using the user name **user02**. Again, you are prompted by the remote system to authenticate with that user's password.

```
[user01@host ~]$ ssh user02@remotehost  
user02@remotehost's password: shadowman  
...output omitted...  
[user02@remotehost ~]$
```

This **ssh** command would run the **hostname** command on the **remotehost** remote system as the **user02** user without accessing the remote interactive shell.

```
[user01@host ~]$ ssh user02@remotehost hostname
user02@remotehost's password: shadowman
remotehost.lab.example.com
[user01@host ~]$
```

Notice that the preceding command displayed the output in the local system's terminal.

Identifying Remote Users

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.

```
[user01@host ~]$ ssh user01@remotehost
user01@remotehost's password: redhat
[user01@remotehost ~]$ w
16:13:38 up 36 min, 1 user, load average: 0.00, 0.00, 0.00
USER      TTY      FROM          LOGIN@    IDLE     JCPU    PCPU WHAT
user02    pts/0    172.25.250.10  16:13     7:30    0.01s  0.01s -bash
user01    pts/1    172.25.250.10  16:24     3.00s  0.01s  0.00s w
[user02@remotehost ~]$
```

The preceding output shows that the **user02** user has logged in to the system on the pseudo-terminal **0** at **16:13** today from the host with the **172.25.250.10** IP address, and has been idle at a shell prompt for seven minutes and thirty seconds. The preceding output also shows that the **user01** user has logged in to the system on the pseudo-terminal **1** and has been idle since last three seconds after executing the **w** command.

SSH host keys

SSH secures communication through public-key encryption. When an SSH client connects to an SSH server, the server sends a copy of its public key to the client before the client logs in. This is used to set up the secure encryption for the communication channel and to authenticate the server to the client.

When a user uses the **ssh** command to connect to an SSH server, the command checks to see if it has a copy of the public key for that server in its local known hosts files. The system administrator may have pre-configured it in **/etc/ssh/ssh_known_hosts**, or the user may have a **~/.ssh/known_hosts** file in their home directory that contains the key.

If the client has a copy of the key, **ssh** will compare the key from the known hosts files for that server to the one it received. If the keys do not match, the client assumes that the network traffic to the server could be hijacked or that the server has been compromised, and seeks the user's confirmation on whether or not to continue with the connection.



Note

Set the **StrictHostKeyChecking** parameter to **yes** in the user-specific **~/.ssh/config** file or the system-wide **/etc/ssh/ssh_config** to cause the **ssh** command to always abort the SSH connection if the public keys do not match.

If the client does not have a copy of the public key in its known hosts files, the **ssh** command will ask you if you want to log in anyway. If you do, a copy of the public key will be saved in your

`~/.ssh/known_hosts` file so that the server's identity can be automatically confirmed in the future.

```
[user01@host ~]$ ssh newhost
The authenticity of host 'remotehost (172.25.250.12)' can't be established.
ECDSA key fingerprint is SHA256:qaS0PToLrqIc02XGkLA0iY7CaP7aPKimerDoaUkv720.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added 'newhost,172.25.250.12' (ECDSA) to the list of known
hosts.
user01@newhost's password: redhat
...output omitted...
[user01@newhost ~]$
```

SSH Known Hosts Key Management

If a server's public key is changed because the key was lost due to hard drive failure, or replaced for some legitimate reason, you will need to edit the known hosts files to make sure the entry for the old public key is replaced with an entry with the new public key in order to log in without errors.

Public keys are stored in the `/etc/ssh/ssh_known_hosts` and each users' `~/.ssh/known_hosts` file on the SSH client. Each key is on one line. The first field is a list of hostnames and IP addresses that share that public key. The second field is the encryption algorithm for the key. The last field is the key itself.

```
[user01@host ~]$ cat ~/.ssh/known_hosts
remotehost,172.25.250.11 ecdsa-sha2-nistp256
AAAAE2VjZHNhLXNoYTItbmlzdHAyNTYAAAAIbmldHdHAyNTYAAABBB0sEi0e+FlaNT6jul8Ag5Nj
+RViZl0yE2w6iYUr+1fPt0IF0Ea0gFZ1LXM37VFTxdgFxHS3D5WhnIfb+68zf8+w=
```

Each remote SSH server that you connect to stores its public key in the `/etc/ssh` directory in files with the extension `.pub`.

```
[user01@remotehost ~]$ ls /etc/ssh/*key.pub
/etc/ssh/ssh_host_ecdsa_key.pub  /etc/ssh/ssh_host_ed25519_key.pub  /etc/ssh/
ssh_host_rsa_key.pub
```



Note

It is a good practice to add entries matching a server's `ssh_host_*key.pub` files to your `~/.ssh/known_hosts` file or the system-wide `/etc/ssh/ssh_known_hosts` file.



References

ssh(1), w(1), and hostname(1) man pages

For more information refer to the *Using secure communications between two systems with OpenSSH* chapter in the *Red Hat Enterprise Linux 8 Securing networks Guide* at

https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/8/html-single/securing_networks/index#using-secure-communications-between-two-systems-with-openssh_securing-networks

► Guided Exercise

Accessing the Remote Command Line

In this exercise, you will log into a remote system as different users and execute commands.

Outcomes

You should be able to:

- Log in to a remote system.
- Execute commands with the OpenSSH secure shell.

Before You Begin

Log in to **workstation** as **student** using **student** as the password.

On **workstation**, run **lab ssh-access start** to start the exercise. This script ensures that the environment is setup correctly.

```
[student@workstation ~]$ lab ssh-access start
```

- 1. From **workstation**, open an SSH session to **servera** as **student**.

```
[student@workstation ~]$ ssh student@servera
...output omitted...
[student@servera ~]$
```

- 2. Open an SSH session to **serverb** as **student**. Accept the host key. Use **student** as the password when prompted for the password of the **student** user on **serverb**.

```
[student@servera ~]$ ssh student@serverb
The authenticity of host 'serverb (172.25.250.11)' can't be established.
ECDSA key fingerprint is SHA256:ERTdjoo0IrIwVSZQnqD5or+JbXfidg0udb3DXBuHWzA.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added 'serverb,172.25.250.11' (ECDSA) to the list of known
hosts.
student@serverb's password: student
...output omitted...
[student@serverb ~]$
```

The host key is recorded in the **/home/student/.ssh/known_hosts** file on **servera** to identify **serverb** because the **student** user has initiated the SSH connection from **servera**. If the **/home/student/.ssh/known_hosts** file does not already exist, it comes into existence as a new file along with the new entry in it. The **ssh** command fails to execute properly if the remote host appears to have a different key than the recorded key.

- 3. Run the **w** command to display the users that are currently logged in to **serverb**.

```
[student@serverb ~]$ w
18:49:29 up 2:55, 1 user,  load average: 0.00, 0.00, 0.00
USER      TTY      FROM          LOGIN@    IDLE     JCPU     PCPU WHAT
student   pts/0    172.25.250.10  18:33     0.00s  0.01s  0.00s w
```

The preceding output indicates that the **student** user has logged in to the system from the host with an IP address of **172.25.250.10** which is **servera** in the classroom network.



Note

The IP address of a system identifies the system on a network. You are going to learn about IP addresses in the later chapter.

- ▶ 4. Exit the **student** user's shell on **serverb**.

```
[student@serverb ~]$ exit
logout
Connection to serverb closed.
[student@servera ~]$
```

- ▶ 5. Open an SSH session to **serverb** as **root**. Use **redhat** as the password of the **root** user.

```
[student@servera ~]$ ssh root@serverb
root@serverb's password: redhat
...output omitted...
[root@serverb ~]#
```

Notice that the preceding **ssh** command did not ask you to accept the host key because it was found among the known hosts. Should the identity of **serverb** change at any time, OpenSSH prompts you to verify and accept the new host key.

- ▶ 6. Run the **w** command to display the users that are currently logged in to **serverb**.

```
[root@serverb ~]# w
19:10:28 up 3:16, 1 user,  load average: 0.00, 0.00, 0.00
USER      TTY      FROM          LOGIN@    IDLE     JCPU     PCPU WHAT
root      pts/0    172.25.250.10  19:09     1.00s  0.01s  0.00s w
```

The preceding output indicates that the **root** user has logged in to the system from the host with an IP address of **172.25.250.10** which is **servera** in the classroom network.

- ▶ 7. Exit the **root** user's shell on **serverb**.

```
[root@serverb ~]# exit
logout
Connection to serverb closed.
[student@servera ~]$
```

- 8. Remove the `/home/student/.ssh/known_hosts` file from `servera`. This causes `ssh` to lose the recorded identities of the remote systems.

```
[student@servera ~]$ rm /home/student/.ssh/known_hosts
```

Host keys can change for legitimate reasons: perhaps the remote machine was replaced because of a hardware failure, or perhaps the remote machine was reinstalled. Usually, it is advisable only to remove the key entry for the particular host in the `known_hosts` file. Since this particular `known_hosts` file has only one entry, you can remove the entire file.

- 9. Open an SSH session to `serverb` as `student`. Accept the host key if asked. Use `student` as the password when prompted for the password of the `student` user on `serverb`.

```
[student@servera ~]$ ssh student@serverb
The authenticity of host 'serverb (172.25.250.11)' can't be established.
ECDSA key fingerprint is SHA256:ERTdjoo0IrIwVSZQnqD5or+JbXfidg0udb3DXBuHWzA.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added 'serverb,172.25.250.11' (ECDSA) to the list of known
hosts.
student@serverb's password: student
...output omitted...
[student@serverb ~]$
```

Notice that the `ssh` command asked for your confirmation to accept or reject the host key because it could not find one for the remote host.

- 10. Exit the `student` user's shell on `serverb` and confirm that a new instance of `known_hosts` exists on `servera`.

```
[student@serverb ~]$ exit
logout
Connection to serverb closed.
[student@servera ~]$ ls -l /home/student/.ssh/known_hosts
-rw-r--r--. 1 student student 183 Feb 1 20:26 /home/student/.ssh/known_hosts
```

- 11. Confirm that the new instance of `known_hosts` file has the host key of `serverb`.

```
[student@servera ~]$ cat /home/student/.ssh/known_hosts
serverb,172.25.250.11 ecdsa-sha2-nistp256
AAAAE2VjZHNhLXNoYTItbmlzdHAyNTYAAAIbmlzdHAyNTYAAABBI9LEYEhwmU1rNqnbBPukH2Ba0/
QBAt9WbS4m03B3MIhhXWKFFNa/UlnjY8NDpEM+hkJe/GmnkcEYMLbCfd9nMA=
```

Actual output will vary.

- 12. Run `hostname` remotely on `serverb` without accessing the interactive shell.

```
[student@servera ~]$ ssh student@serverb hostname
student@serverb's password: student
serverb.lab.example.com
```

The preceding command displayed the full hostname of the remote system `serverb`.

- 13. Exit the **student** user's shell on **servera**.

```
[student@servera ~]$ exit  
logout  
Connection to servera closed.
```

Finish

On **workstation**, run **lab ssh-access finish** to complete this exercise.

```
[student@workstation ~]$ lab ssh-access finish
```

This concludes the guided exercise.

Configuring SSH Key-based Authentication

Objectives

After completing this section, you should be able to configure a user account to use key-based authentication to log in to remote systems securely without a password.

SSH Key-based Authentication

You can configure an SSH server to allow you to authenticate without a password by using key-based authentication. This is based on a private-public key scheme.

To do this, you generate a matched pair of cryptographic key files. One is a private key, the other a matching 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 connect to, and is used to verify the private key. The public key does not need to be secret.

You put a copy of the public key in your account on the server. When you try to log in, the SSH server can use the public key to issue a challenge that can only be correctly answered by using the private key. As a result, your **ssh** client can automatically authenticate your login to the server with your unique copy of the private key. This allows you to securely access systems in a way that doesn't require you to enter a password interactively every time.

Generating SSH Keys

To create a private key and matching public key for authentication, use the **ssh-keygen** command. By default, your private and public keys are saved in your `~/.ssh/id_rsa` and `~/.ssh/id_rsa.pub` files, respectively.

```
[user@host ~]$ ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/home/user/.ssh/id_rsa): Enter
Created directory '/home/user/.ssh'.
Enter passphrase (empty for no passphrase): Enter
Enter same passphrase again: Enter
Your identification has been saved in /home/user/.ssh/id_rsa.
Your public key has been saved in /home/user/.ssh/id_rsa.pub.
The key fingerprint is:
SHA256:vxutUNPi03QDCyvkYm1oIx35hmMrHpPKWFdIYu3HV+w user@host.lab.example.com
The key's randomart image is:
+---[RSA 2048]---+
|                               |
|                               |
|                               |
|                               |
| . . . . |
| o o o . |
| . = o o . |
| o + = S E . |
| ..o o + * + |
| .+% o . + B . |
```

```
|=*o0 . . + *      |
|++. . +.      |
+---[SHA256]-----+
```

If you do not specify a passphrase when **ssh-keygen** prompts you, the generated private key is not protected. In this case, anyone with your private key file could use it for authentication. If you set a passphrase, then you will need to enter that passphrase when you use the private key for authentication. (Therefore, you would be using the private key's passphrase rather than your password on the remote host to authenticate.)

You can run a helper program called **ssh-agent** which can temporarily cache your private key passphrase in memory at the start of your session to get true passwordless authentication. This will be discussed later in this section.

The following example of the **ssh-keygen** command shows the creation of the passphrase-protected private key alongside the public key.

```
[user@host ~]$ ssh-keygen -f .ssh/key-with-pass
Generating public/private rsa key pair.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in .ssh/key-with-pass.
Your public key has been saved in .ssh/key-with-pass.pub.
The key fingerprint is:
SHA256:w3GGB7EyHUr4a0cNPKmhNKS7dl1YsMVLvFZJ77VxAo user@host.lab.example.com
The key's randomart image is:
+---[RSA 2048]----+
|   . + =.o ... |
|   = B XEo o. |
|   . o O X =.... |
|   = = = B = o. |
|= + * * S . |
|.+= o + . |
| + . |
| |
| |
+---[SHA256]-----+
```

The **-f** option with the **ssh-keygen** command determines the files where the keys are saved. In the preceding example, the private and public keys are saved in the **/home/user/.ssh/key-with-pass** / **/home/user/.ssh/key-with-pass.pub** files, respectively.



Warning

During further SSH keypair generation, unless you specify a unique file name, you are prompted for permission to overwrite the existing **id_rsa** and **id_rsa.pub** files. If you overwrite the existing **id_rsa** and **id_rsa.pub** files, then you must replace the old public key with the new one on all the SSH servers that have your old public key.

Once the SSH keys have been generated, they are stored by default in the **.ssh/** directory of the user's home directory. The permission modes must be 600 on the private key and 644 on the public key.

Sharing the Public Key

Before key-based authentication can be used, the public key needs to be copied to the destination system. The **ssh-copy-id** command copies the public key of the SSH keypair to the destination system. If you omit the path to the public key file while running **ssh-copy-id**, it uses the default **/home/user/.ssh/id_rsa.pub** file.

```
[user@host ~]$ ssh-copy-id -i .ssh/key-with-pass.pub user@remotehost
/usr/bin/ssh-copy-id: INFO: Source of key(s) to be installed: "/home/user/.ssh/
id_rsa.pub"
/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
user@remotehost's password: redhat
Number of key(s) added: 1

Now try logging into the machine, with: "ssh 'user@remotehost'"
and check to make sure that only the key(s) you wanted were added.
```

After the public key is successfully transferred to a remote system, you can authenticate to the remote system using the corresponding private key while logging in to the remote system over SSH. If you omit the path to the private key file while running the **ssh** command, it uses the default **/home/user/.ssh/id_rsa** file.

```
[user@host ~]$ ssh -i .ssh/key-with-pass user@remotehost
Enter passphrase for key '.ssh/key-with-pass': redhatpass
...output omitted...
[user@remotehost ~]$ exit
logout
Connection to remotehost closed.
[user@host ~]$
```

Using ssh-agent for Non-interactive Authentication

If your SSH private key is protected with a passphrase, you normally have to enter the passphrase to use the private key for authentication. However, you can use a program called **ssh-agent** to temporarily cache the passphrase in memory. Then any time that you use SSH to log in to another system with the private key, **ssh-agent** will automatically provide the passphrase for you. This is convenient, and can improve security by providing fewer opportunities for someone "shoulder surfing" to see you type the passphrase in.

Depending on your local system's configuration, if you initially log in to the GNOME graphical desktop environment, the **ssh-agent** program might automatically be started and configured for you.

If you log in on a text console, log in using **ssh**, or use **sudo** or **su**, you will probably need to start **ssh-agent** manually for that session. You can do this with the following command:

```
[user@host ~]$ eval $(ssh-agent)
Agent pid 10155
[user@host ~]$
```

**Note**

When you run **ssh-agent**, it prints out some shell commands. You need to run these commands to set environment variables used by programs like **ssh-add** to communicate with it. The **eval \$(ssh-agent)** command starts **ssh-agent** and runs those commands to automatically set those environment variables for that shell session. It also displays the PID of the **ssh-agent** process.

Once **ssh-agent** is running, you need to tell it the passphrase for your private key or keys. You can do this with the **ssh-add** command.

The following **ssh-add** commands add the private keys from **/home/user/.ssh/id_rsa** (the default) and **/home/user/.ssh/key-with-pass** files, respectively.

```
[user@host ~]$ ssh-add
Identity added: /home/user/.ssh/id_rsa (user@host.lab.example.com)
[user@host ~]$ ssh-add .ssh/key-with-pass
Enter passphrase for .ssh/key-with-pass: redhatpass
Identity added: .ssh/key-with-pass (user@host.lab.example.com)
```

After successfully adding the private keys to the **ssh-agent** process, you can invoke an SSH connection using the **ssh** command. If you are using any private key file other than the default **/home/user/.ssh/id_rsa** file, then you must use the **-i** option with the **ssh** command to specify the path to the private key file.

The following example of the **ssh** command uses the default private key file to authenticate to an SSH server.

```
[user@host ~]$ ssh user@remotehost
Last login: Fri Apr  5 10:53:50 2019 from host.example.com
[user@remotehost ~]$
```

The following example of the **ssh** command uses the **/home/user/.ssh/key-with-pass** (non-default) private key file to authenticate to an SSH server. The private key in the following example has already been decrypted and added to its parent **ssh-agent** process, so the **ssh** command does not prompt you to decrypt the private key by interactively entering its passphrase.

```
[user@host ~]$ ssh -i .ssh/key-with-pass user@remotehost
Last login: Mon Apr  8 09:44:20 2019 from host.example.com
[user@remotehost ~]$
```

When you log out of the session that started **ssh-agent**, the process will exit and your the passphrases for your private keys will be cleared from memory.

**References**

ssh-keygen(1), **ssh-copy-id(1)**, **ssh-agent(1)**, **ssh-add(1)** man pages

► Guided Exercise

Configuring SSH Key-based Authentication

In this exercise, you will configure a user to use key-based authentication for SSH.

Outcomes

You should be able to:

- Generate an SSH key pair without passphrase protection.
- Generate an SSH key pair with passphrase protection.
- Authenticate using both passphrase-less and passphrase-protected SSH keys.

Before You Begin

Log in to **workstation** as **student** using **student** as the password.

On **workstation**, run **lab ssh-configure start** to start the exercise. This script creates the necessary user accounts.

```
[student@workstation ~]$ lab ssh-configure start
```

- 1. From **workstation**, open an SSH session to **serverb** as **student**.

```
[student@workstation ~]$ ssh student@serverb  
...output omitted...  
[student@serverb ~]$
```

- 2. Use the **su** command to switch to the **operator1** user on **serverb**. Use **redhat** as the password of **operator1**.

```
[student@serverb ~]$ su - operator1  
Password: redhat  
[operator1@serverb ~]$
```

- 3. Use the **ssh-keygen** command to generate SSH keys. Do not enter a passphrase.

```
[operator1@serverb ~]$ ssh-keygen  
Generating public/private rsa key pair.  
Enter file in which to save the key (/home/operator1/.ssh/id_rsa): Enter  
Created directory '/home/operator1/.ssh'.  
Enter passphrase (empty for no passphrase): Enter  
Enter same passphrase again: Enter  
Your identification has been saved in /home/operator1/.ssh/id_rsa.  
Your public key has been saved in /home/operator1/.ssh/id_rsa.pub.
```

```
The key fingerprint is:  
SHA256:JainiQdnRosC+xXh0qsJQQLzBNULdb+jJbyrCZQBERI  
operator1@serverb.lab.example.com  
The key's randomart image is:  
+---[RSA 2048]---+  
|E+*+ooo . . . . |  
|.= o.o o . . . |  
|o.. = . . o . . |  
|+. + * . o . . |  
|+ = X . S + . . |  
| + @ + = . . . |  
|. + = o . . . . |  
|.o . . . . . . . |  
|o . . . . . . . |  
+---[SHA256]---+
```

- ▶ 4. Use the **ssh-copy-id** command to send the public key of the SSH key pair to **operator1** on **servera**. Use **redhat** as the password of **operator1** on **servera**.

```
[operator1@serverb ~]$ ssh-copy-id operator1@servera  
/usr/bin/ssh-copy-id: INFO: Source of key(s) to be installed: "/home/  
operator1/.ssh/id_rsa.pub"  
The authenticity of host 'servera (172.25.250.10)' can't be established.  
ECDSA key fingerprint is SHA256:ERTdjoo0IrIWVszQnqD5or+JbXfidg0udb3DXBuHWzA.  
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  
operator1@servera's password: redhat  
Number of key(s) added: 1  
  
Now try logging into the machine, with: "ssh 'operator1@servera'"  
and check to make sure that only the key(s) you wanted were added.
```

- ▶ 5. Execute the **hostname** command on **servera** remotely using SSH without accessing the remote interactive shell.

```
[operator1@serverb ~]$ ssh operator1@servera hostname  
servera.lab.example.com
```

Notice that the preceding **ssh** command did not prompt you for a password because it used the passphrase-less private key against the exported public key to authenticate as **operator1** on **servera**. This approach is not secure, because anyone who has access to the private key file can log in to **servera** as **operator1**. The secure alternative is to protect the private key with a passphrase, which is the next step.

- 6. Use the **ssh-keygen** command to generate another set of SSH keys with passphrase-protection. Save the key as **/home/operator1/.ssh/key2**. Use **redhatpass** as the passphrase of the private key.

**Warning**

If you do not specify the file where the key gets saved, the default file (**/home/user/.ssh/id_rsa**) is used. You have already used the default file name when generating SSH keys in the preceding step, so it is vital that you specify a non-default file, otherwise the existing SSH keys will be overwritten.

```
[operator1@serverb ~]$ ssh-keygen -f .ssh/key2
Generating public/private rsa key pair.
Enter passphrase (empty for no passphrase): redhatpass
Enter same passphrase again: redhatpass
Your identification has been saved in .ssh/key2.
Your public key has been saved in .ssh/key2.pub.
The key fingerprint is:
SHA256:0OctCjfPm5QrbPBgqbEIwCcw5AI4oSlMEbgLrBQ1HwKI
operator1@serverb.lab.example.com
The key's randomart image is:
+---[RSA 2048]---+
|O=X*           |
|OB=.          |
|E*o.         |
|Booo .       |
|..= . o S    |
|+.o  o      |
|+.oo+ o     |
|+o.O.+     |
|+ . =o.    |
+---[SHA256]---
```

- 7. Use the **ssh-copy-id** command to send the public key of the passphrase-protected key pair to **operator1** on **servera**.

```
[operator1@serverb ~]$ ssh-copy-id -i .ssh/key2.pub operator1@servera
/usr/bin/ssh-copy-id: INFO: Source of key(s) to be installed: ".ssh/key2.pub"
/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
```

Number of key(s) added: 1

Now try logging into the machine, with: "ssh 'operator1@servera'"
and check to make sure that only the key(s) you wanted were added.

Notice that the preceding **ssh-copy-id** command did not prompt you for a password because it used the public key of the passphrase-less private key that you exported to **servera** in the preceding step.

- 8. Execute the **hostname** command on **servera** remotely with SSH without accessing the remote interactive shell. Use **/home/operator1/.ssh/key2** as the identity file. Specify **redhatpass** as the passphrase, which you set for the private key in the preceding step.

```
[operator1@serverb ~]$ ssh -i .ssh/key2 operator1@servera hostname  
Enter passphrase for key '.ssh/key2': redhatpass  
servera.lab.example.com
```

Notice that the preceding **ssh** command prompted you for the passphrase you used to protect the private key of the SSH key pair. This passphrase protects the private key. Should an attacker gain access to the private key, the attacker cannot use it to access other systems because the private key itself is protected with a passphrase. The **ssh** command uses a different passphrase than the one for **operator1** on **servera**, requiring users to know both.

You can use **ssh-agent**, as in the following step, to avoid interactively typing in the passphrase while logging in with SSH. Using **ssh-agent** is both more convenient and more secure in situations where the administrators log in to remote systems regularly.

- 9. Run **ssh-agent** in your Bash shell and add the passphrase-protected private key (**/home/operator1/.ssh/key2**) of the SSH key pair to the shell session.

```
[operator1@serverb ~]$ eval $(ssh-agent)  
Agent pid 21032  
[operator1@serverb ~]$ ssh-add .ssh/key2  
Enter passphrase for .ssh/key2: redhatpass  
Identity added: .ssh/key2 (operator1@serverb.lab.example.com)
```

The preceding **eval** command started **ssh-agent** and configured this shell session to use it. You then used **ssh-add** to provide the unlocked private key to **ssh-agent**.

- 10. Execute the **hostname** command on **servera** remotely without accessing a remote interactive shell. Use **/home/operator1/.ssh/key2** as the identity file.

```
[operator1@serverb ~]$ ssh -i .ssh/key2 operator1@servera hostname  
servera.lab.example.com
```

Notice that the preceding **ssh** command did not prompt you to enter the passphrase interactively.

- 11. Open another terminal on **workstation** and open an SSH session to **serverb** as **student**.

```
[student@workstation ~]$ ssh student@serverb  
...output omitted...  
[student@serverb ~]$
```

- 12. On **serverb**, use the **su** command to switch to **operator1** and invoke an SSH connection to **servera**. Use **/home/operator1/.ssh/key2** as the identity file to authenticate using the SSH keys.

- 12.1. Use the **su** command to switch to **operator1**. Use **redhat** as the password of **operator1**.

```
[student@serverb ~]$ su - operator1  
Password: redhat  
[operator1@serverb ~]$
```

12.2. Open an SSH session to **servera** as **operator1**.

```
[operator1@serverb ~]$ ssh -i .ssh/key2 operator1@servera  
Enter passphrase for key '.ssh/key2': redhatpass  
...output omitted...  
[operator1@servera ~]$
```

Notice that the preceding **ssh** command prompted you to enter the passphrase interactively because you did not invoke the SSH connection from the shell that you used to start **ssh-agent**.

► 13. Exit all the shells you are using in the second terminal.

13.1. Log out of **servera**.

```
[operator1@servera ~]$ exit  
logout  
Connection to servera closed.  
[operator1@serverb ~]$
```

13.2. Exit the **operator1** and **student** shells on **serverb** to return to the **student** user's shell on **workstation**.

```
[operator1@serverb ~]$ exit  
logout  
[student@serverb ~]$ exit  
logout  
Connection to serverb closed.  
[student@workstation ~]$
```

13.3. Close the second terminal on **workstation**.

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

► 14. Log out of **serverb** on the first terminal and conclude this exercise.

14.1. From the first terminal, exit the **operator1** user's shell on **serverb**.

```
[operator1@serverb ~]$ exit  
logout  
[student@serverb ~]$
```

The **exit** command caused you to exit the **operator1** user's shell, terminating the shell session where **ssh-agent** was active, and return to the **student** user's shell on **serverb**.

14.2. Exit the **student** user's shell on **serverb** to return to the **student** user's shell on **workstation**.

```
[student@serverb ~]$ exit  
logout  
Connection to serverb closed.  
[student@workstation ~]$
```

Finish

On **workstation**, run **lab ssh-configure finish** to complete this exercise.

```
[student@workstation ~]$ lab ssh-configure finish
```

This concludes the guided exercise.

Customizing OpenSSH Service Configuration

Objectives

After completing this section, you should be able to restrict direct logins as **root** and disable password-based authentication for the OpenSSH service.

Configuring the OpenSSH Server

OpenSSH service is provided by a daemon called **sshd**. Its main configuration file is **/etc/ssh/sshd_config**.

The default configuration of the OpenSSH server works well. However, you might want to make some changes to strengthen the security of your system. There are two common changes you might want to make. You might want to prohibit direct remote login to the **root** account, and you might want to prohibit password-based authentication (in favor of SSH private key authentication).

Prohibit the Superuser From Logging in Using SSH

It is a good practice to prohibit direct login to the **root** user account from remote systems. Some of the risks of allowing direct login as **root** include:

- The user name **root** exists on every Linux system by default, so a potential attacker only has to guess the password, instead of a valid user name and password combination. This reduces complexity for an attacker.
- The **root** user has unrestricted privileges, so its compromise can lead to maximum damage to the system.
- From an auditing perspective, it can be hard to track which authorized user logged in as **root** and made changes. If users have to log in as a regular user and switch to the **root** account, this generates a log event that can be used to help provide accountability.

The OpenSSH server uses the **PermitRootLogin** configuration setting in the **/etc/ssh/sshd_config** configuration file to allow or prohibit users logging in to the system as **root**.

```
PermitRootLogin yes
```

With the **PermitRootLogin** parameter to **yes**, as it is by default, people are permitted to log in as **root**. To prevent this, set the value to **no**. Alternatively, to prevent password-based authentication but allow private key-based authentication for **root**, set the **PermitRootLogin** parameter to **without-password**.

The SSH server (**sshd**) must be reloaded for any changes to take effect.

```
[root@host ~]# systemctl reload sshd
```

Prohibiting Password-Based Authentication for SSH

Allowing only private key-based logins to the remote command line has various advantages:

- Attackers cannot use password guessing attacks to remotely break into known accounts on the system.
- With passphrase-protected private keys, an attacker needs both the passphrase and a copy of the private key. With passwords, an attacker just needs the password.
- By using passphrase-protected private keys in conjunction with **ssh-agent**, the passphrase is exposed less frequently since it is entered less frequently, and logging in is more convenient for the user.

The OpenSSH server uses the **PasswordAuthentication** parameter in the **/etc/ssh/sshd_config** configuration file to control whether users can use password-based authentication to log in to the system.

```
>PasswordAuthentication yes
```

The default value of **yes** for the **PasswordAuthentication** parameter in the **/etc/ssh/sshd_config** configuration file causes the SSH server to allow users to use password-based authentication while logging in. The value of **no** for **PasswordAuthentication** prevents users from using password-based authentication.

Keep in mind that whenever you change the **/etc/ssh/sshd_config** file, you must reload the **sshd** service for changes to take effect.



Important

Remember, if you turn off password-based authentication for **ssh**, you need to have a way to ensure that the user's **~/.ssh/authorized_keys** file on the remote server is populated with their public key, so that they can log in.



References

ssh(1), **sshd_config(5)** man pages

► Guided Exercise

Customizing OpenSSH Service Configuration

In this exercise, you will disable direct logins as **root** and password-based authentication for the OpenSSH service on one of your servers.

Outcomes

You should be able to:

- Disable direct logins as **root** over **ssh**.
- Disable password-based authentication for remote users to connect over SSH.

Before You Begin

Log in to **workstation** as **student** using **student** as the password.

On **workstation**, run **lab ssh-customize start** to start the exercise. This script creates the necessary user accounts and files.

```
[student@workstation ~]$ lab ssh-customize start
```

- 1. From **workstation**, open an SSH session to **serverb** as **student**.

```
[student@workstation ~]$ ssh student@serverb
...output omitted...
[student@serverb ~]$
```

- 2. Use the **su** command to switch to **operator2** on **serverb**. Use **redhat** as the password of **operator2**.

```
[student@serverb ~]$ su - operator2
Password: redhat
[operator2@serverb ~]$
```

- 3. Use the **ssh-keygen** command to generate SSH keys. Do not enter any passphrase for the keys.

```
[operator2@serverb ~]$ ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/home/operator2/.ssh/id_rsa): Enter
Created directory '/home/operator2/.ssh'.
Enter passphrase (empty for no passphrase): Enter
Enter same passphrase again: Enter
Your identification has been saved in /home/operator2/.ssh/id_rsa.
Your public key has been saved in /home/operator2/.ssh/id_rsa.pub.
```

```
The key fingerprint is:  
SHA256:JainiQdnRosC+xXh0qsJQLzBNULdb+jJbyrCZQBERI  
operator1@serverb.lab.example.com  
The key's randomart image is:  
+---[RSA 2048]---+  
|E+*+ooo . . . . |  
|.= o.o o . . . |  
|o.. = . . o . . |  
|+. + * . o . . |  
|+ = X . S + . . |  
| + @ + = . . . |  
|. + = o . . . . |  
|.o . . . . . . . |  
|o . . . . . . . |  
+---[SHA256]---+
```

- ▶ 4. Use the **ssh-copy-id** command to send the public key of the SSH key pair to **operator2** on **servera**. Use **redhat** as the password of **operator2** on **servera**.

```
[operator2@serverb ~]$ ssh-copy-id operator2@servera  
/usr/bin/ssh-copy-id: INFO: Source of key(s) to be installed: "/home/  
operator1/.ssh/id_rsa.pub"  
The authenticity of host 'servera (172.25.250.10)' can't be established.  
ECDSA key fingerprint is SHA256:ERTdjoo0IrIWVSZQnqD5or+JbXfidg0udb3DXBuHWzA.  
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  
operator2@servera's password: redhat  
Number of key(s) added: 1  
  
Now try logging into the machine, with: "ssh 'operator2@servera'"  
and check to make sure that only the key(s) you wanted were added.
```

- ▶ 5. Confirm that you can successfully log in to **servera** as **operator2** using the SSH keys.

5.1. Open an SSH session to **servera** as **operator2**.

```
[operator2@serverb ~]$ ssh operator2@servera  
...output omitted...  
[operator2@servera ~]$
```

Notice that the preceding **ssh** command used SSH keys for authentication.

5.2. Log out of **servera**.

```
[operator2@servera ~]$ exit  
logout  
Connection to servera closed.
```

- ▶ 6. Confirm that you can successfully log in to **servera** as **root** using **redhat** as the password.

- 6.1. Open an SSH session to **servera** as **root** using **redhat** as the password.

```
[operator2@serverb ~]$ ssh root@servera  
root@servera's password: redhat  
...output omitted...  
[root@servera ~]#
```

Notice that the preceding **ssh** command used the password of the superuser for authentication because SSH keys do not exist for the superuser.

- 6.2. Log out of **servera**.

```
[root@servera ~]# exit  
logout  
Connection to servera closed.  
[operator2@serverb ~]$
```

- 7. Confirm that you can successfully log in to **servera** as **operator3** using **redhat** as the password.

- 7.1. Open an SSH session to **servera** as **operator3** using **redhat** as the password.

```
[operator2@serverb ~]$ ssh operator3@servera  
operator3@servera's password: redhat  
...output omitted...  
[operator3@servera ~]$
```

Notice that the preceding **ssh** command used the password of **operator3** for authentication because SSH keys do not exist for **operator3**.

- 7.2. Log out of **servera**.

```
[operator3@servera ~]# exit  
logout  
Connection to servera closed.  
[operator2@serverb ~]$
```

- 8. Configure **sshd** on **servera** to prevent users logging in as **root**. Use **redhat** as the password of the superuser when required.

- 8.1. Open an SSH session to **servera** as **operator2** using the SSH keys.

```
[operator2@serverb ~]$ ssh operator2@servera  
...output omitted...  
[operator2@servera ~]$
```

- 8.2. On **servera**, switch to **root**. Use **redhat** as the password of the **root** user.

```
[operator2@servera ~]$ su -  
Password: redhat  
[root@servera ~]#
```

- 8.3. Set **PermitRootLogin** to **no** in **/etc/ssh/sshd_config** and reload **sshd**. You may use **vim /etc/ssh/sshd_config** to edit the configuration file of **sshd**.

```
...output omitted...
PermitRootLogin no
...output omitted...
[root@servera ~]# systemctl reload sshd
```

- 8.4. Open another terminal on **workstation** and open an SSH session to **serverb** as **operator2**. From **serverb**, try logging in to **servera** as **root**. This should fail because you disabled **root** user login over SSH in the preceding step.



Note

For your convenience, password-less login is already configured between **workstation** and **serverb** in the classroom environment.

```
[student@workstation ~]$ ssh operator2@serverb
...output omitted...
[operator2@serverb ~]$ ssh root@servera
root@servera's password: redhat
Permission denied, please try again.
root@servera's password: redhat
Permission denied, please try again.
root@servera's password: redhat
root@servera: Permission denied (publickey,gssapi-keyex,gssapi-with-mic,password).
```

By default, the **ssh** command attempts to authenticate using key-based authentication first and then, if that fails, password-based authentication.

- 9. Configure **sshd** on **servera** to allow users to authenticate using SSH keys only, rather than their passwords.
- 9.1. Return to the first terminal that has the **root** user's shell active on **servera**. Set **PasswordAuthentication** to **no** in **/etc/ssh/sshd_config** and reload **sshd**. You may use **vim /etc/ssh/sshd_config** to edit the configuration file of **sshd**.

```
...output omitted...
PasswordAuthentication no
...output omitted...
[root@servera ~]# systemctl reload sshd
```

- 9.2. Go to the second terminal that has **operator2** user's shell active on **serverb** and try logging in to **servera** as **operator3**. This should fail because SSH keys are not configured for **operator3**, and the **sshd** service on **servera** does not allow the use of passwords for authentication.

```
[operator2@serverb ~]$ ssh operator3@servera
operator3@servera: Permission denied (publickey,gssapi-keyex,gssapi-with-mic).
```

**Note**

For more granularity, you may use the explicit **-o PubkeyAuthentication=no** and **-o PasswordAuthentication=yes** options with the **ssh** command. This allows you to override the **ssh** command's defaults and confidently determine that the preceding command fails based on the settings you adjusted in **/etc/ssh/sshd_config** in the preceding step.

- 9.3. Return to the first terminal that has the **root** user's shell active on **servera**. Verify that **PubkeyAuthentication** is enabled in **/etc/ssh/sshd_config**. You may use **vim /etc/ssh/sshd_config** to view the configuration file of **sshd**.

```
...output omitted...
#PubkeyAuthentication yes
...output omitted...
```

Notice that the **PubkeyAuthentication** line is commented. Any commented line in this file uses the default value. Commented lines indicate the default values of a parameter. The public key authentication of SSH is active by default, as the commented line indicates.

- 9.4. Return to the second terminal that has **operator2** user's shell active on **serverb** and try logging in to **servera** as **operator2**. This should succeed because the SSH keys are configured for **operator2** to log in to **servera** from **serverb**.

```
[operator2@serverb ~]$ ssh operator2@servera
...output omitted...
[operator2@servera ~]$
```

- 9.5. From the second terminal, exit the **operator2** user's shell on both **servera** and **serverb**.

```
[operator2@servera ~]$ exit
logout
Connection to servera closed.
[operator2@serverb ~]$ exit
logout
Connection to serverb closed.
[student@workstation ~]$
```

- 9.6. Close the second terminal on **workstation**.

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

- 9.7. From the first terminal, exit the **root** user's shell on **servera**.

```
[root@servera ~]# exit
logout
```

- 9.8. From the first terminal, exit the **operator2** user's shell on both **servera** and **serverb**.

```
[operator2@servera ~]$ exit  
logout  
Connection to servera closed.  
[operator2@serverb ~]$ exit  
logout  
[student@serverb ~]$
```

9.9. Log out of **serverb** and return to the **student** user's shell on **workstation**.

```
[student@serverb ~]$ exit  
logout  
Connection to serverb closed.  
[student@workstation ~]$
```

Finish

On **workstation**, run **lab ssh-customize finish** to complete this exercise.

```
[student@workstation ~]$ lab ssh-customize finish
```

This concludes the guided exercise.

▶ Lab

Configuring and Securing SSH

Performance Checklist

In this lab, you will set up key-based authentication for users, and disable direct login as **root** and password authentication for all users for the OpenSSH service on one of your servers.

Outcomes

You should be able to:

- Authenticate using SSH keys.
- Prevent users from directly logging in as **root** over **ssh**.
- Prevent users from logging in to the system using SSH password-based authentication.

Before You Begin

Log in to **workstation** as **student** using **student** as the password.

On **workstation**, run **lab ssh-review start** to start the exercise. This script creates the necessary user accounts and files.

```
[student@workstation ~]$ lab ssh-review start
```

1. From **workstation**, open an SSH session to **servera** as **student**.
2. Use the **su** command to switch to **production1** on **servera**.
3. Use the **ssh-keygen** command to generate passphrase-less SSH keys for **production1** on **servera**.
4. Use the **ssh-copy-id** command to send the public key of the SSH key pair to **production1** on **serverb**.
5. Confirm that **production1** can successfully log in to **serverb** using the SSH keys.
6. Configure **sshd** on **serverb** to prevent users logging in as **root**. Use **redhat** as the password of the superuser.
7. Configure **sshd** on **serverb** to allow users to authenticate using SSH keys only, rather than their passwords.

Evaluation

On **workstation**, run the **lab ssh-review grade** command to confirm success of this exercise.

```
[student@workstation ~]$ lab ssh-review grade
```

Finish

On **workstation**, run **lab ssh-review finish** to complete this lab.

```
[student@workstation ~]$ lab ssh-review finish
```

This concludes the lab.

► Solution

Configuring and Securing SSH

Performance Checklist

In this lab, you will set up key-based authentication for users, and disable direct login as **root** and password authentication for all users for the OpenSSH service on one of your servers.

Outcomes

You should be able to:

- Authenticate using SSH keys.
- Prevent users from directly logging in as **root** over **ssh**.
- Prevent users from logging in to the system using SSH password-based authentication.

Before You Begin

Log in to **workstation** as **student** using **student** as the password.

On **workstation**, run **lab ssh-review start** to start the exercise. This script creates the necessary user accounts and files.

```
[student@workstation ~]$ lab ssh-review start
```

- From **workstation**, open an SSH session to **servera** as **student**.

```
[student@workstation ~]$ ssh student@servera
...output omitted...
[student@servera ~]$
```

- Use the **su** command to switch to **production1** on **servera**.

```
[student@servera ~]$ su - production1
Password: redhat
[production1@servera ~]$
```

- Use the **ssh-keygen** command to generate passphrase-less SSH keys for **production1** on **servera**.

```
[production1@servera ~]$ ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/home/production1/.ssh/id_rsa): Enter
Created directory '/home/production1/.ssh'.
Enter passphrase (empty for no passphrase): Enter
Enter same passphrase again: Enter
Your identification has been saved in /home/production1/.ssh/id_rsa.
```

```
Your public key has been saved in /home/production1/.ssh/id_rsa.pub.
The key fingerprint is:
SHA256:CsWCAmW0g5qaJuJLzIAcengNj3u21kbrPP4Ysl3PXCA
production1@servera.lab.example.com
The key's randomart image is:
+---[RSA 2048]----+
|..o          |
|o+ . .       |
|= o . o     |
|.+  o        |
|o.. . E .    |
|*o.= .... . |
|Xo+ +oo...   |
|Oo .+==+ + . |
| *o+o=*o. +  |
+---[SHA256]----+
```

4. Use the **ssh-copy-id** command to send the public key of the SSH key pair to **production1** on **serverb**.

```
[production1@servera ~]$ ssh-copy-id production1@serverb
/usr/bin/ssh-copy-id: INFO: Source of key(s) to be installed: "/home/
production1/.ssh/id_rsa.pub"
The authenticity of host 'serverb (172.25.250.11)' can't be established.
ECDSA key fingerprint is SHA256:ERTdjoo0IrIwVSZQnqD5or+JbXfidg0udb3DXBuHWzA.
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
production1@serverb's password: redhat
Number of key(s) added: 1

Now try logging into the machine, with: "ssh 'production1@serverb'"
and check to make sure that only the key(s) you wanted were added.
```

5. Confirm that **production1** can successfully log in to **serverb** using the SSH keys.

```
[production1@servera ~]$ ssh production1@serverb
...output omitted...
[production1@serverb ~]$
```

6. Configure **sshd** on **serverb** to prevent users logging in as **root**. Use **redhat** as the password of the superuser.

- 6.1. Use the **su -** command to switch to **root** on **serverb**.

```
[production1@serverb ~]$ su -
Password: redhat
[root@serverb ~]#
```

- 6.2. Set **PermitRootLogin** to **no** in **/etc/ssh/sshd_config** and reload **sshd**. You may use **vim /etc/ssh/sshd_config** to edit the configuration file of **sshd**.

```
...output omitted...
PermitRootLogin no
...output omitted...
[root@serverb ~]# systemctl reload sshd.service
```

- 6.3. Open another terminal on **workstation** and open an SSH session to **servera** as **production1**. From **servera**, try logging in to **serverb** as **root**. This should fail because you disabled **root** user login over SSH in the preceding step.

**Note**

For your convenience, the password-less login is already configured between **workstation** and **servera** in the classroom environment.

```
[student@workstation ~]$ ssh production1@servera
...output omitted...
[production1@servera ~]$ ssh root@serverb
root@serverb's password: redhat
Permission denied, please try again.
root@serverb's password: redhat
Permission denied, please try again.
root@serverb's password: redhat
root@serverb: Permission denied (publickey,gssapi-keyex,gssapi-with-mic,password).
[production1@servera ~]$
```

The preceding **ssh** command returned after three failed attempts to log in to **servera** as **root**. By default, the **ssh** command prefers to use SSH keys for authentication but if it does not find the necessary keys of the user, it requests the user's password for authentication.

7. Configure **sshd** on **serverb** to allow users to authenticate using SSH keys only, rather than their passwords.

- 7.1. Return to the first terminal that has the **root** user's shell active on **serverb**. Set **PasswordAuthentication** to **no** in **/etc/ssh/sshd_config** and reload **sshd**. You may use **vim /etc/ssh/sshd_config** to edit the configuration file of **sshd**.

```
...output omitted...
PasswordAuthentication no
...output omitted...
[root@serverb ~]# systemctl reload sshd
```

- 7.2. Go to the second terminal that has **production1** user's shell active on **servera** and try logging in to **serverb** as **production2**. This should fail because SSH keys are not configured for **production2**, and the **sshd** service on **serverb** does not allow the use of passwords for authentication.

```
[production1@servera ~]$ ssh production2@serverb
production2@serverb: Permission denied (publickey,gssapi-keyex,gssapi-with-mic).
```

**Note**

For more granularity, you may use the explicit **-o PubkeyAuthentication=no** and **-o PasswordAuthentication=yes** options with the **ssh** command. This allows you to override the **ssh** command's defaults and confidently establish whether the preceding command actually fails based on the settings you adjusted in **/etc/ssh/sshd_config** in the preceding step.

- 7.3. Return to the first terminal that has the **root** user's shell active on **serverb**. Verify that **PubkeyAuthentication** is enabled in **/etc/ssh/sshd_config**. You may use **vim /etc/ssh/sshd_config** to view the configuration file of **sshd**.

```
...output omitted...
#PubkeyAuthentication yes
...output omitted...
```

Notice that the **PubkeyAuthentication** line is commented. Any commented line in this file uses the default value. Commented lines indicate the default values of a parameter. The public key authentication of SSH is active by default, as the commented line indicates.

- 7.4. Return to the second terminal that has **production1** user's shell active on **servera** and try logging in to **serverb** as **production1**. This should succeed because SSH keys are configured for **production1** to log in to **serverb** from **servera**.

```
[production1@servera ~]$ ssh production1@serverb
...output omitted...
[production1@serverb ~]$
```

- 7.5. From the second terminal, exit the **production1** user's shell on both **serverb** and **servera**.

```
[production1@serverb ~]$ exit
logout
Connection to serverb closed.
[production1@servera ~]$ exit
logout
[student@workstation ~]$
```

- 7.6. Close the second terminal on **workstation**.

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

- 7.7. From the first terminal, exit the **root** user's shell on **serverb**.

```
[root@serverb ~]# exit
logout
```

- 7.8. From the first terminal, exit the **production1** user's shell on both **serverb** and **servera**.

```
[production1@serverb ~]$ exit  
logout  
Connection to serverb closed.  
[production1@servera ~]$ exit  
logout  
[student@servera ~]$
```

7.9. Log out of **servera** and return to the **student** user's shell on **workstation**.

```
[student@servera ~]$ exit  
logout  
Connection to servera closed.  
[student@workstation ~]$
```

Evaluation

On **workstation**, run the **lab ssh-review grade** command to confirm success of this exercise.

```
[student@workstation ~]$ lab ssh-review grade
```

Finish

On **workstation**, run **lab ssh-review finish** to complete this lab.

```
[student@workstation ~]$ lab ssh-review finish
```

This concludes the lab.

Summary

In this chapter, you learned:

- The **ssh** command allows users to access remote systems securely using the SSH protocol.
- A client system stores remote servers' identities in `~/.ssh/known_hosts` and `/etc/ssh/ssh_known_hosts`.
- SSH supports both password-based and key-based authentication.
- The **ssh-keygen** command generates an SSH key pair for authentication. The **ssh-copy-id** command exports the public key to remote systems.
- The **sshd** service implements the SSH protocol on Red Hat Enterprise Linux systems.
- It is a recommended practice to configure **sshd** to disable remote logins as **root** and to require public key authentication rather than password-based authentication.

Chapter 11

Analyzing and Storing Logs

Goal

Locate and accurately interpret logs of system events for troubleshooting purposes.

Objectives

- Describe the basic logging architecture used by Red Hat Enterprise Linux to record events.
- Interpret events in relevant syslog files to troubleshoot problems or review system status.
- Find and interpret entries in the system journal to troubleshoot problems or review system status.
- Configure the system journal to preserve the record of events when a server is rebooted.
- Maintain accurate time synchronization using NTP and configure the time zone to ensure correct time stamps for events recorded by the system journal and logs.

Sections

- Describing System Log Architecture (and Quiz)
- Reviewing Syslog Files (and Guided Exercise)
- Reviewing System Journal Entries (and Guided Exercise)
- Preserving the System Journal (and Guided Exercise)
- Maintaining Accurate Time (and Guided Exercise)

Lab

Analyzing and Storing Logs

Describing System Log Architecture

Objectives

After completing this section, you should be able to describe the basic logging architecture used by Red Hat Enterprise Linux to record events.

System Logging

Processes and the operating system kernel record a log of events that happen. These logs are used to audit the system and troubleshoot problems.

Many systems record logs of events in text files which are kept in the **/var/log** directory. These logs can be inspected using normal text utilities such as **less** and **tail**.

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. The **systemd-journald** and **rsyslog** services handle the syslog messages in Red Hat Enterprise Linux 8.

The **systemd-journald** service is at the heart of the operating system event logging architecture. It collects event messages from many sources including the kernel, output from the early stages of the boot process, standard output and standard error from daemons as they start up and run, and syslog events. It then restructures them into a standard format, and writes them into a structured, indexed system journal. By default, this journal is stored on a file system that does not persist across reboots.

However, the **rsyslog** service reads syslog messages received by **systemd-journald** from the journal as they arrive. It then processes the syslog events, recording them to its log files or forwarding them to other services according to its own configuration.

The **rsyslog** service sorts and writes syslog messages to the log files that do persist across reboots in **/var/log**. The **rsyslog** service sorts the log messages to specific log files based on the type of program that sent each message, or *facility*, and the priority of each syslog message.

In addition to syslog message files, the **/var/log** directory contains log files from other services on the system. The following table lists some useful files in the **/var/log** directory.

Selected System Log Files

Log file	Type of Messages Stored
/var/log/messages	Most syslog messages are logged here. Exceptions include messages related to authentication and email processing, scheduled job execution, and those which are purely debugging-related.
/var/log/secure	Syslog messages related to security and authentication events.
/var/log/maillog	Syslog messages related to the mail server.
/var/log/cron	Syslog messages related to scheduled job execution.

Log file	Type of Messages Stored
<code>/var/log/boot.log</code>	Non-syslog console messages related to system startup.

**Note**

Some applications do not use syslog to manage their log messages, although typically, they do place their log files in a subdirectory of /var/log. For example, the Apache Web Server saves log messages to files in a subdirectory of the `/var/log` directory.

**References**

`systemd-journald.service(8)`, `rsyslogd(8)`, and `rsyslog.conf(5)` man pages

For more information refer to the *Troubleshooting problems using log files* section in the *Red Hat Enterprise Linux 8 Configuring basic system settings Guide* at https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/8/html-single/configuring_basic_system_settings/index#troubleshooting-problems-using-log-files_getting-started-with-system-administration

► Quiz

Describing System Log Architecture

Choose the correct answer to the following questions:

- ▶ 1. Which of these log files stores most syslog messages, with the exception of those that are related to authentication, mail, scheduled jobs, and debugging?
 - a. `/var/log/maillog`
 - b. `/var/log/boot.log`
 - c. `/var/log/messages`
 - d. `/var/log/secure`
- ▶ 2. Which log file stores syslog messages related to security and authentication operations in the system?
 - a. `/var/log/maillog`
 - b. `/var/log/boot.log`
 - c. `/var/log/messages`
 - d. `/var/log/secure`
- ▶ 3. Which service sorts and organizes syslog messages into files in `/var/log`?
 - a. `rsyslog`
 - b. `systemd-journald`
 - c. `auditd`
 - d. `tuned`
- ▶ 4. Which directory accommodates the human-readable syslog files?
 - a. `/sys/kernel/debug`
 - b. `/var/log/journal`
 - c. `/run/log/journal`
 - d. `/var/log`
- ▶ 5. Which file stores syslog messages related to the mail server?
 - a. `/var/log/lastlog`
 - b. `/var/log/maillog`
 - c. `/var/log/tallylog`
 - d. `/var/log/boot.log`

► **6. Which file stores syslog messages related to the scheduled jobs?**

- a. **/var/log/cron**
- b. **/var/log/tallylog**
- c. **/var/log/spooler**
- d. **/var/log/secure**

► **7. What file stores console messages related to system startup?**

- a. **/var/log/messages**
- b. **/var/log/cron**
- c. **/var/log/boot.log**
- d. **/var/log/secure**

► Solution

Describing System Log Architecture

Choose the correct answer to the following questions:

- ▶ 1. Which of these log files stores most syslog messages, with the exception of those that are related to authentication, mail, scheduled jobs, and debugging?
 - a. /var/log/maillog
 - b. /var/log/boot.log
 - c. /var/log/messages
 - d. /var/log/secure

- ▶ 2. Which log file stores syslog messages related to security and authentication operations in the system?
 - a. /var/log/maillog
 - b. /var/log/boot.log
 - c. /var/log/messages
 - d. /var/log/secure

- ▶ 3. Which service sorts and organizes syslog messages into files in /var/log?
 - a. rsyslog
 - b. systemd-journald
 - c. auditd
 - d. tuned

- ▶ 4. Which directory accommodates the human-readable syslog files?
 - a. /sys/kernel/debug
 - b. /var/log/journal
 - c. /run/log/journal
 - d. /var/log

- ▶ 5. Which file stores syslog messages related to the mail server?
 - a. /var/log/lastlog
 - b. /var/log/maillog
 - c. /var/log/tallylog
 - d. /var/log/boot.log

► **6. Which file stores syslog messages related to the scheduled jobs?**

- a. `/var/log/cron`
- b. `/var/log/tallylog`
- c. `/var/log/spooler`
- d. `/var/log/secure`

► **7. What file stores console messages related to system startup?**

- a. `/var/log/messages`
- b. `/var/log/cron`
- c. `/var/log/boot.log`
- d. `/var/log/secure`

Reviewing Syslog Files

Objectives

After completing this section, you should be able to interpret events in relevant syslog files to troubleshoot problems or review system status.

Logging Events to the System

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). Available facilities are documented in the **rsyslog.conf(5)** man page.

The following table lists the standard eight syslog priorities from highest to lowest.

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 **rsyslog** service uses the facility and priority of log messages to determine how to handle them. This is configured by rules in the **/etc/rsyslog.conf** file and any file in the **/etc/rsyslog.d** directory that has a file name extension of **.conf**. Software packages can easily add rules by installing an appropriate file in the **/etc/rsyslog.d** directory.

Each rule that controls how to sort syslog messages is a line in one of the configuration files. The left side of each line indicates the facility and severity of the syslog messages the rule matches. The right side of each line indicates what file to save the log message in (or where else to deliver the message). An asterisk (*) is a wildcard that matches all values.

For example, the following line would record messages sent to the **authpriv** facility at any priority to the file **/var/log/secure**:

```
authpriv.*          /var/log/secure
```

Log messages sometimes match more than one rule in **rsyslog.conf**. In such cases, one message is stored in more than one log file. To limit messages stored, the key word **none** in the priority field indicates that no messages for the indicated facility should be stored in the given file.

Instead of logging syslog messages to a file, they can also be printed to the terminals of all logged-in users. The **rsyslog.conf** file has a setting to print all the syslog messages with the **emerg** priority to the terminals of all logged-in users.

Sample Rules of Rsyslog

```
##### 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 syslog subsystem has many more features beyond the scope of this course. For those who wish to explore further, consult the **rsyslog.conf(5)** man page and the extensive HTML documentation in **/usr/share/doc/rsyslog/html/index.html** contained in the **rsyslog-doc** package, available from the AppStream repository in Red Hat Enterprise Linux 8.

Log File Rotation

The **logrotate** tool rotates log files to keep them from taking up too much space in the file system containing the **/var/log** directory. When a log file is rotated, it is renamed with an extension indicating the date it was rotated. For example, the old **/var/log/messages** file may

become **/var/log/messages-20190130** if it is rotated on 2019-01-30. 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 oldest log file is discarded to free disk space. A scheduled 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.

Analyzing a Syslog Entry

Log messages start with the oldest message on top and the newest message at the end of the log file. The **rsyslog** service uses a standard format while recording entries in log files. The following example explains the anatomy of a log 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 name and PID number that sent the log message
- ④ The actual message sent

Monitoring Logs

Monitoring one or more log files for events is helpful to reproduce problems and issues. The **tail -f /path/to/file** command outputs the last 10 lines of the file specified and continues to output new lines in the file as they get written.

For example, to monitor for failed login attempts, run the **tail** command in one terminal and then in another terminal, run the **ssh** command as the **root** user while a user tries to log in to the system.

In the first terminal, run the following **tail** command:

```
[root@host ~]# tail -f /var/log/secure
```

In the second terminal, run the following **ssh** command:

```
[root@host ~]# ssh root@localhost  
root@localhost's password: redhat  
...output omitted...  
[root@host ~]#
```

Return to the first terminal and view the logs.

```
...output omitted...  
Feb 10 09:01:13 host sshd[2712]: Accepted password for root from 172.25.254.254  
port 56801 ssh2  
Feb 10 09:01:13 host sshd[2712]: pam_unix(sshd:session): session opened for user  
root by (uid=0)
```

Sending Syslog Messages Manually

The **logger** command can send messages to the **rsyslog** service. By default, it sends the message to the **user** facility with the **notice** priority (**user.notice**) unless specified otherwise with the **-p** option. It is useful to test any change to the **rsyslog** service configuration.

To send a message to the **rsyslog** service that gets recorded in the **/var/log/boot.log** log file, execute the following **logger** command:

```
[root@host ~]# logger -p local7.notice "Log entry created on host"
```



References

logger(1), **tail(1)**, **rsyslog.conf(5)**, and **logrotate(8)** man pages

rsyslog Manual

- **/usr/share/doc/rsyslog/html/index.html** provided by the *rsyslog-doc* package

For more information refer to the *Troubleshooting problems using log files* section in the *Red Hat Enterprise Linux 8 Configuring basic system settings Guide* at https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/8/html-single/configuring_basic_system_settings/index#troubleshooting-problems-using-log-files_getting-started-with-system-administration

► Guided Exercise

Reviewing Syslog Files

In this exercise, you will reconfigure **rsyslog** to write specific log messages to a new file.

Outcomes

You should be able to configure the **rsyslog** service to write all log messages with the **debug** priority to the **/var/log/messages-debug** log file.

Before You Begin

Log in to **workstation** as **student** using **student** as the password.

On **workstation**, run **lab log-configure start** to start the exercise. This script ensures that the environment is setup correctly.

```
[student@workstation ~]$ lab log-configure start
```

- 1. From **workstation**, open an SSH session to **servera** as **student**.

```
[student@workstation ~]$ ssh student@servera
...output omitted...
[student@servera ~]$
```

- 2. Configure **rsyslog** on **servera** to log all messages with the **debug** priority, or higher, for any service into the new **/var/log/messages-debug** log file by adding the **rsyslog** configuration file **/etc/rsyslog.d/debug.conf**.
 - 2.1. Use the **sudo -i** command to switch to the **root** user. Specify **student** as the password for the **student** user if asked while running the **sudo -i** command.

```
[student@servera ~]$ sudo -i
[sudo] password for student: student
[root@servera ~]#
```

- 2.2. Create the **/etc/rsyslog.d/debug.conf** file with the necessary entries to redirect all log messages having the **debug** priority to **/var/log/messages-debug**. You may use the **vim /etc/rsyslog.d/debug.conf** command to create the file with the following content.

```
* .debug /var/log/messages-debug
```

This configuration line catches syslog messages with any facility and a **debug** or above priority level. The **rsyslog** service will write the matching messages to the **/var/log/messages-debug** file. The wildcard (*) in the **facility** or **priority** fields of the configuration line indicates any facility or priority.

- 2.3. Restart the **rsyslog** service.

```
[root@servera ~]# systemctl restart rsyslog
```

- ▶ 3. Verify that all the log messages with the **debug** priority appears in the **/var/log/messages-debug** file.
- 3.1. Use the **logger** command with the **-p** option to generate a log message with the **user** facility and the **debug** priority.

```
[root@servera ~]# logger -p user.debug "Debug Message Test"
```

- 3.2. Use the **tail** command to view the last ten log messages from the **/var/log/messages-debug** file and confirm that you see the **Debug Message Test** message among the other log messages.

```
[root@servera ~]# tail /var/log/messages-debug
Feb 13 18:22:38 servera systemd[1]: Stopping System Logging Service...
Feb 13 18:22:38 servera rsyslogd[25176]: [origin software="rsyslogd"
  swVersion="8.37.0-9.el8" x-pid="25176" x-info="http://www.rsyslog.com"] exiting
  on signal 15.
Feb 13 18:22:38 servera systemd[1]: Stopped System Logging Service.
Feb 13 18:22:38 servera systemd[1]: Starting System Logging Service...
Feb 13 18:22:38 servera rsyslogd[25410]: environment variable TZ is not set, auto
  correcting this to TZ=/etc/localtime [v8.37.0-9.el8 try http://www.rsyslog.com/
  e/2442 ]
Feb 13 18:22:38 servera systemd[1]: Started System Logging Service.
Feb 13 18:22:38 servera rsyslogd[25410]: [origin software="rsyslogd"
  swVersion="8.37.0-9.el8" x-pid="25410" x-info="http://www.rsyslog.com"] start
Feb 13 18:27:58 servera student[25416]: Debug Message Test
```

- 3.3. Exit both the **root** and **student** users' shells on **servera** to return to the **student** user's shell on **workstation**.

```
[root@servera ~]# exit
logout
[student@servera ~]$ exit
logout
Connection to servera closed.
[student@workstation ~]$
```

Finish

On **workstation**, run **lab log-configure finish** to complete this exercise. This script ensures that the environment is restored back to the clean state.

```
[student@workstation ~]$ lab log-configure finish
```

This concludes the guided exercise.

Reviewing System Journal Entries

Objectives

After completing this section, you should be able to find and interpret entries in the system journal to troubleshoot problems or review system status.

Finding Events

The **systemd-journald** service stores logging data in a structured, indexed binary file called the journal. This data includes extra information about the log event. For example, for syslog events this includes the facility and the priority of the original message.



Important

In Red Hat Enterprise Linux 8, the **/run/log** directory stores the system journal by default. The contents of the **/run/log** directory get cleared after a reboot. You can change this setting, and how to do so is discussed later in this chapter.

To retrieve log messages from the journal, use the **journalctl** command. You can use this command to view all messages in the journal, or to search for specific events based on a wide range of options and criteria. If you run the command as **root**, you have full access to the journal. Regular users can also use this command, but might be restricted from seeing certain messages.

```
[root@host ~]# journalctl
...output omitted...
Feb 21 17:46:25 host.lab.example.com systemd[24263]: Stopped target Sockets.
Feb 21 17:46:25 host.lab.example.com systemd[24263]: Closed D-Bus User Message Bus
Socket.
Feb 21 17:46:25 host.lab.example.com systemd[24263]: Closed Multimedia System.
Feb 21 17:46:25 host.lab.example.com systemd[24263]: Reached target Shutdown.
Feb 21 17:46:25 host.lab.example.com systemd[24263]: Starting Exit the Session...
Feb 21 17:46:25 host.lab.example.com systemd[24268]: pam_unix(systemd-
user:session): session c>
Feb 21 17:46:25 host.lab.example.com systemd[1]: Stopped User Manager for UID
1001.
Feb 21 17:46:25 host.lab.example.com systemd[1]: user-runtime-dir@1001.service:
Unit not needed
Feb 21 17:46:25 host.lab.example.com systemd[1]: Stopping /run/user/1001 mount
wrapper...
Feb 21 17:46:25 host.lab.example.com systemd[1]: Removed slice User Slice of UID
1001.
Feb 21 17:46:25 host.lab.example.com systemd[1]: Stopped /run/user/1001 mount
wrapper.
Feb 21 17:46:36 host.lab.example.com sshd[24434]: Accepted publickey for root from
172.25.250.>
Feb 21 17:46:37 host.lab.example.com systemd[1]: Started Session 20 of user root.
Feb 21 17:46:37 host.lab.example.com systemd-logind[708]: New session 20 of user
root.
```

Chapter 11 | Analyzing and Storing Logs

```
Feb 21 17:46:37 host.lab.example.com sshd[24434]: pam_unix(sshd:session): session opened for u>
Feb 21 18:01:01 host.lab.example.com CROND[24468]: (root) CMD (run-parts /etc/cron.hourly)
Feb 21 18:01:01 host.lab.example.com run-parts[24471]: (/etc/cron.hourly) starting
  @anacron
Feb 21 18:01:01 host.lab.example.com run-parts[24477]: (/etc/cron.hourly) finished
  @anacron
lines 1464-1487/1487 (END) q
```

The **journalctl** command highlights important log messages: messages at **notice** or **warning** priority are in bold text while messages at the **error** priority or higher are in red text.

The key to successfully using the journal for troubleshooting and auditing is to limit journal searches to show only relevant output.

By default, **journalctl -n** shows the last 10 log entries. You can adjust this with an optional argument that specifies how many log entries to display. For the last five log entries, run the following **journalctl** command:

```
[root@host ~]# journalctl -n 5
-- Logs begin at Wed 2019-02-20 16:01:17 +07, end at Thu 2019-02-21 18:01:01 +07.
--
...output omitted...
Feb 21 17:46:37 host.lab.example.com systemd-logind[708]: New session 20 of user root.
Feb 21 17:46:37 host.lab.example.com sshd[24434]: pam_unix(sshd:session): session opened for u>
Feb 21 18:01:01 host.lab.example.com CROND[24468]: (root) CMD (run-parts /etc/cron.hourly)
Feb 21 18:01:01 host.lab.example.com run-parts[24471]: (/etc/cron.hourly) starting
  @anacron
Feb 21 18:01:01 host.lab.example.com run-parts[24477]: (/etc/cron.hourly) finished
  @anacron
lines 1-6/6 (END) q
```

Similar to the **tail -f** command, the **journalctl -f** command outputs the last 10 lines of the system journal and continues to output new journal entries as they get written to the journal. To exit the **journalctl -f** process, use the **Ctrl+C** key combination.

```
[root@host ~]# journalctl -f
-- Logs begin at Wed 2019-02-20 16:01:17 +07. --
...output omitted...
Feb 21 18:01:01 host.lab.example.com run-parts[24477]: (/etc/cron.hourly) finished
  @anacron
Feb 21 18:22:42 host.lab.example.com sshd[24437]: Received disconnect from
  172.25.250.250 port 48710:11: disconnected by user
Feb 21 18:22:42 host.lab.example.com sshd[24437]: Disconnected from user root
  172.25.250.250 port 48710
Feb 21 18:22:42 host.lab.example.com sshd[24434]: pam_unix(sshd:session): session closed for user root
Feb 21 18:22:42 host.lab.example.com systemd-logind[708]: Session 20 logged out.
  Waiting for processes to exit.
Feb 21 18:22:42 host.lab.example.com systemd-logind[708]: Removed session 20.
```

Chapter 11 | Analyzing and Storing Logs

```
Feb 21 18:22:43 host.lab.example.com sshd[24499]: Accepted
publickey for root from 172.25.250.250 port 48714 ssh2: RSA
SHA256:1UGybTe52L2jzEJa1HLVKn9UCKrTv3ZxnmJol1Fro
Feb 21 18:22:44 host.lab.example.com systemd-logind[708]: New session 21 of user
root.
Feb 21 18:22:44 host.lab.example.com systemd[1]: Started Session 21 of user root.
Feb 21 18:22:44 host.lab.example.com sshd[24499]: pam_unix(sshd:session): session
opened for user root by (uid=0)
^C
[root@host ~]#
```

To help troubleshoot problems, you might want to filter the output of the journal based on the priority of the journal entries. The **journalctl -p** takes either the name or the number of a priority level and shows the journal entries for entries at that priority and above. The **journalctl** command understands the **debug, info, notice, warning, err, crit, alert, and emerg** priority levels.

Run the following **journalctl** command to list journal entries at the **err** priority or higher:

```
[root@host ~]# journalctl -p err
-- Logs begin at Wed 2019-02-20 16:01:17 +07, end at Thu 2019-02-21 18:01:01 +07.
--
...output omitted...
Feb 20 16:01:17 host.lab.example.com kernel: Detected CPU family 6 model 13
stepping 3
Feb 20 16:01:17 host.lab.example.com kernel: Warning: Intel Processor - this
hardware has not undergone testing by Red Hat and might not be certif>
Feb 20 16:01:20 host.lab.example.com smartd[669]: DEVICESCAN failed: glob(3)
aborted matching pattern /dev/discs/disc*
Feb 20 16:01:20 host.lab.example.com smartd[669]: In the system's table of devices
NO devices found to scan
lines 1-5/5 (END) q
```

When looking for specific events, you can 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 argument in the format "YYYY-MM-DD hh:mm:ss" (the double-quotes are required to preserve the space in the option). If the date is omitted, the command assumes the current day, and if the time is omitted, the command assumes the whole day starting at 00:00:00. Both options take **yesterday, today, and tomorrow** as valid arguments in addition to the date and time field.

Run the following **journalctl** command to list all journal entries from today's records.

```
[root@host ~]# journalctl --since today
-- Logs begin at Wed 2019-02-20 16:01:17 +07, end at Thu 2019-02-21 18:31:14 +07.
--
...output omitted...
Feb 21 18:22:44 host.lab.example.com systemd-logind[708]: New session 21 of user
root.
Feb 21 18:22:44 host.lab.example.com systemd[1]: Started Session 21 of user root.
Feb 21 18:22:44 host.lab.example.com sshd[24499]: pam_unix(sshd:session): session
opened for user root by (uid=0)
Feb 21 18:31:13 host.lab.example.com systemd[1]: Starting dnf makecache...
```

```
Feb 21 18:31:14 host.lab.example.com dnf[24533]: Red Hat Enterprise Linux 8.0
  AppStream (dvd)      637 kB/s | 2.8 kB      00:00
Feb 21 18:31:14 host.lab.example.com dnf[24533]: Red Hat Enterprise Linux 8.0
  BaseOS (dvd)        795 kB/s | 2.7 kB      00:00
Feb 21 18:31:14 host.lab.example.com dnf[24533]: Metadata cache created.
Feb 21 18:31:14 host.lab.example.com systemd[1]: Started dnf makecache.
lines 533-569/569 (END) q
```

Run the following `journalctl` command to list all journal entries ranging from **2019-02-10 20:30:00** to **2019-02-13 12:00:00**.

```
[root@host ~]# journalctl --since "2019-02-10 20:30:00" \
--until "2019-02-13 12:00:00"
...output omitted...
```

You can also specify all entries since a time relative to the present. For example, to specify all entries in the last hour, you can use the following command:

```
[root@host ~]# journalctl --since "-1 hour"
...output omitted...
```



Note

You can use other, more sophisticated time specifications with the `--since` and `--until` options. For some examples, see the **systemd.time(7)** man page.

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. Any displayed extra field 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@host ~]# journalctl -o verbose
-- Logs begin at Wed 2019-02-20 16:01:17 +07, end at Thu 2019-02-21 18:31:14 +07.
--
...output omitted...
Thu 2019-02-21 18:31:14.509128 +07...
    _BOOT_ID=4409bbf54680496d94e090de9e4a9e23
    _MACHINE_ID=73ab164e278e48be9bf80e80714a8cd5
    _SYSLOG_FACILITY=3
    _SYSLOG_IDENTIFIER=systemd
    _UID=0
    _GID=0
    CODE_FILE=../src/core/job.c
    CODE_LINE=826
    CODE_FUNC=job_log_status_message
    JOB_TYPE=start
    JOB_RESULT=done
    MESSAGE_ID=39f53479d3a045ac8e11786248231fbf
    _TRANSPORT=journal
    _PID=1
    COMM=systemd
```

```
_EXE=/usr/lib/systemd/systemd
_CMDLINE=/usr/lib/systemd/systemd --switched-root --system --deserialize 18
_CAP_EFFECTIVE=3fffffff
_SELINUX_CONTEXT=system_u:system_r:init_t:s0
_SYSTEMD_CGROUP=/init.scope
_SYSTEMD_UNIT=init.scope
_SYSTEMD_SLICE=-.slice
UNIT=dnf-makecache.service
MESSAGE=Started dnf makecache.
_HOSTNAME=host.lab.example.com
INVOCATION_ID=d6f90184663f4309835a3e8ab647cb0e
_SOURCE_REALTIME_TIMESTAMP=1550748674509128
lines 32239-32275/32275 (END) q
```

The following list gives the common fields of the system journal that can be used to search for lines relevant to a particular process or event.

- `_COMM` is the name of the command
- `_EXE` is the path to the executable for the process
- `_PID` is the PID of the process
- `_UID` is the UID of the user running the process
- `_SYSTEMD_UNIT` is the systemd unit that started the process

More than one of the system journal fields can be combined to form a granular search query with the **journalctl** command. For example, the following **journalctl** command shows all journal entries related to the **sshd.service systemd** unit from a process with PID 1182.

```
[root@host ~]# journalctl _SYSTEMD_UNIT=sshd.service _PID=1182
Apr 03 19:34:27 host.lab.example.com sshd[1182]: Accepted password for root
      from ::1 port 52778 ssh2
Apr 03 19:34:28 host.lab.example.com sshd[1182]: pam_unix(sshd:session): session
      opened for user root by (uid=0)
...output omitted...
```



Note

For a list of commonly used journal fields, consult the **systemd.journal-fields(7)** man page.



References

journalctl(1), **systemd.journal-fields(7)**, and **systemd.time(7)** man pages

For more information refer to the *Troubleshooting problems using log files* section in the *Red Hat Enterprise Linux 8 Configuring basic system settings Guide* at https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/8/html-single/configuring_basic_system_settings/index#troubleshooting-problems-using-log-files_getting-started-with-system-administration

► Guided Exercise

Reviewing System Journal Entries

In this exercise, you will search the system journal for entries recording events that match specific criteria.

Outcomes

You should be able to search the system journal for entries recording events based on different criteria.

Before You Begin

Log in to **workstation** as **student** using **student** as the password.

On **workstation**, run **lab log-query start** to start the exercise. This script ensures that the environment is setup correctly.

```
[student@workstation ~]$ lab log-query start
```

- 1. From **workstation**, open an SSH session to **servera** as **student**.

```
[student@workstation ~]$ ssh student@servera
...output omitted...
[student@servera ~]$
```

- 2. Use the **_PID=1** match with the **journalctl** command to display only log events originating from the **systemd** process running with the process identifier of 1 on **servera**. To quit **journalctl**, press **q**.

```
[student@servera ~]$ journalctl _PID=1
...output omitted...
Feb 13 13:21:08 localhost systemd[1]: Found device /dev/disk/by-uuid/
cdf61ded-534c-4bd6-b458-cab18b1a72ea.
Feb 13 13:21:08 localhost systemd[1]: Started dracut initqueue hook.
Feb 13 13:21:08 localhost systemd[1]: Found device /dev/disk/by-
uuid/44330f15-2f9d-4745-ae2e-20844f22762d.
Feb 13 13:21:08 localhost systemd[1]: Reached target Initrd Root Device.
lines 1-5/5 (END) q
[student@servera ~]$
```



Note

The **journalctl** command may produce a different output on your system.

- 3. Use the **_UID=81** match with the **journalctl** command to display all log events originating from a system service started with the user identifier of 81 on **servera**. To quit **journalctl** press **q**.

```
[student@servera ~]$ journalctl _UID=81
...output omitted...
Feb 22 01:29:09 servera.lab.example.com dbus-daemon[672]: [system] Activating via
systemd: service name='org.freedesktop.nm_dispatcher'>
Feb 22 01:29:09 servera.lab.example.com dbus-daemon[672]: [system] Successfully
activated service 'org.freedesktop.nm_dispatcher'
lines 1-5/5 (END) q
[student@servera ~]$
```

- 4. Use the **-p warning** option with the **journalctl** command to display log events with priority **warning** and above on **servera**. To quit **journalctl** press **q**.

```
[student@servera ~]$ journalctl -p warning
...output omitted...
Feb 13 13:21:07 localhost kernel: Detected CPU family 6 model 13 stepping 3
Feb 13 13:21:07 localhost kernel: Warning: Intel Processor - this hardware has not
undergone testing by Red Hat and might not >
Feb 13 13:21:07 localhost kernel: acpi PNP0A03:00: fail to add MMCONFIG
information, can't access extended PCI configuration s>
Feb 13 13:21:07 localhost rpc.statd[288]: Running as root. chown /var/lib/nfs/
statd to choose different user
Feb 13 13:21:07 localhost rpc.idmapd[293]: Setting log level to 0
...output omitted...
Feb 13 13:21:13 servera.lab.example.com rsyslogd[1172]: environment variable TZ is
not set, auto correcting this to TZ=/etc/lo>
Feb 13 14:51:42 servera.lab.example.com systemd[1]: cgroup compatibility
translation between legacy and unified hierarchy sett>
Feb 13 17:15:37 servera.lab.example.com rsyslogd[25176]: environment variable TZ
is not set, auto correcting this to TZ=/etc/l>
Feb 13 18:22:38 servera.lab.example.com rsyslogd[25410]: environment variable TZ
is not set, auto correcting this to TZ=/etc/l>
Feb 13 18:47:55 servera.lab.example.com rsyslogd[25731]: environment variable TZ
is not set, auto correcting this to TZ=/etc/l>
lines 1-17/17 (END) q
[student@servera ~]$
```

- 5. Display all log events recorded in the past 10 minutes from the current time on **servera**.

- 5.1. Use the **--since** option with the **journalctl** command to display all log events recorded in the past 10 minutes on **servera**. To quit **journalctl** press **q**.

```
[student@servera ~]$ journalctl --since "-10min"
...output omitted...
Feb 13 22:31:01 servera.lab.example.com CROND[25890]: (root) CMD (run-parts /etc/
cron.hourly)
Feb 13 22:31:01 servera.lab.example.com run-parts[25893]: (/etc/cron.hourly)
starting 0anacron
Feb 13 22:31:01 servera.lab.example.com run-parts[25899]: (/etc/cron.hourly)
finished 0anacron
```

```

Feb 13 22:31:41 servera.lab.example.com sshd[25901]: Bad protocol version
identification 'brain' from 172.25.250.254 port 37450
Feb 13 22:31:42 servera.lab.example.com sshd[25902]: Accepted publickey for root
from 172.25.250.254 port 37452 ssh2: RSA SHA256:...
Feb 13 22:31:42 servera.lab.example.com systemd[1]: Started /run/user/0 mount
wrapper.
Feb 13 22:31:42 servera.lab.example.com systemd[1]: Created slice User Slice of
UID 0.
Feb 13 22:31:42 servera.lab.example.com systemd[1]: Starting User Manager for UID
0...
Feb 13 22:31:42 servera.lab.example.com systemd[1]: Started Session 118 of user
root.
Feb 13 22:31:42 servera.lab.example.com systemd-logind[712]: New session 118 of
user root.
Feb 13 22:31:42 servera.lab.example.com systemd[25906]: pam_unix(systemd-
user:session): session opened for user root by (uid=0)
...output omitted...
lines 1-32/84 39% q
[student@servera ~]$
```

- ▶ 6. Use the `--since` option and the `_SYSTEMD_UNIT="sshd.service"` match with the `journalctl` command to display all the log events originating from the `sshd` service recorded since **09:00:00** this morning on `servera`. To quit `journalctl` press `q`.



Note

You may or may not be located in the same timezone as your classroom. Check the time on `servera` and adjust the `--since` value accordingly if required.

```

[student@servera ~]$ journalctl --since 9:00:00 _SYSTEMD_UNIT="sshd.service"
...output omitted...
Feb 13 13:21:12 servera.lab.example.com sshd[727]: Server listening on 0.0.0.0
port 22.
Feb 13 13:21:12 servera.lab.example.com sshd[727]: Server listening on :: port 22.
Feb 13 13:22:07 servera.lab.example.com sshd[1238]: Accepted publickey for student
from 172.25.250.250 port 50590 ssh2: RSA SHA256:...
Feb 13 13:22:07 servera.lab.example.com sshd[1238]: pam_unix(sshd:session):
session opened for user student by (uid=0)
Feb 13 13:22:08 servera.lab.example.com sshd[1238]: pam_unix(sshd:session):
session closed for user student
Feb 13 13:25:47 servera.lab.example.com sshd[1289]: Accepted publickey for root
from 172.25.250.254 port 37194 ssh2: RSA SHA256:...
Feb 13 13:25:47 servera.lab.example.com sshd[1289]: pam_unix(sshd:session):
session opened for user root by (uid=0)
Feb 13 13:25:47 servera.lab.example.com sshd[1289]: pam_unix(sshd:session):
session closed for user root
Feb 13 13:25:48 servera.lab.example.com sshd[1316]: Accepted publickey for root
from 172.25.250.254 port 37196 ssh2: RSA SHA256:...
Feb 13 13:25:48 servera.lab.example.com sshd[1316]: pam_unix(sshd:session):
session opened for user root by (uid=0)
Feb 13 13:25:48 servera.lab.example.com sshd[1316]: pam_unix(sshd:session):
session closed for user root
```

Chapter 11 | Analyzing and Storing Logs

```
Feb 13 13:26:07 servera.lab.example.com sshd[1355]: Accepted publickey for student
from 172.25.250.254 port 37198 ssh2: RSA SH>
Feb 13 13:26:07 servera.lab.example.com sshd[1355]: pam_unix(sshd:session):
session opened for user student by (uid=0)
Feb 13 13:52:28 servera.lab.example.com sshd[1473]: Accepted publickey for root
from 172.25.250.254 port 37218 ssh2: RSA SHA25>
Feb 13 13:52:28 servera.lab.example.com sshd[1473]: pam_unix(sshd:session):
session opened for user root by (uid=0)
...output omitted...
lines 1-32 q
[student@servera ~]$
```

▶ 7. Log out of **servera**.

```
[student@servera ~]$ exit
logout
Connection to servera closed.
[student@workstation ~]$
```

Finish

On **workstation**, run **lab log-query finish** to complete this exercise. This script ensures that the environment is restored back to the clean state.

```
[student@workstation ~]$ lab log-query finish
```

This concludes the guided exercise.

Preserving the System Journal

Objectives

After completing this section, you should be able to configure the system journal to preserve the record of events when a server is rebooted.

Storing the System Journal Permanently

By default, the system journals are kept in the `/run/log/journal` directory, which means the journals are cleared when the system reboots. You can change the configuration settings of the `systemd-journald` service in the `/etc/systemd/journald.conf` file to make the journals persist across reboot.

The **Storage** parameter in the `/etc/systemd/journald.conf` file defines whether to store system journals in a volatile manner or persistently across reboot. Set this parameter to **persistent**, **volatile**, **auto**, or **none** as follows:

- **persistent**: stores journals in the `/var/log/journal` directory which persists across reboots.
If the `/var/log/journal` directory does not exist, the `systemd-journald` service creates it.
- **volatile**: stores journals in the volatile `/run/log/journal` directory.
As the `/run` file system is temporary and exists only in the runtime memory, data stored in it, including system journals, do not persist across a reboot.
- **auto**: if the `/var/log/journal` directory exists, then `systemd-journald` uses persistent storage, otherwise it uses volatile storage.
This is the default action if the **Storage** parameter is not set.
- **none**: do not use any storage. All logs are dropped but log forwarding will still work as expected.

The advantage of persistent system journals is that the historic data is available immediately at boot. However, even with a persistent journal, not all data is kept forever. The journal has a built-in log rotation mechanism that triggers monthly. In addition, by default, the journals are not 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 for both the runtime and persistent journals in `/etc/systemd/journald.conf`. The current limits on the size of the journal are logged when the `systemd-journald` process starts. The following command output shows the journal entries that reflect the current size limits:

```
[user@host ~]$ journalctl | grep -E 'Runtime|System journal'
Feb 25 13:01:46 localhost systemd-journald[147]: Runtime journal (/run/log/
journal/ae06db7da89142138408d77efea9229c) is 8.0M, max 91.4M, 83.4M free.
Feb 25 13:01:48 remotehost.lab.example.com systemd-journald[548]: Runtime journal
(/run/log/journal/73ab164e278e48be9bf80e80714a8cd5) is 8.0M, max 91.4M, 83.4M
free.
Feb 25 13:01:48 remotehost.lab.example.com systemd-journald[548]: System journal
(/var/log/journal/73ab164e278e48be9bf80e80714a8cd5) is 8.0M, max 3.7G, 3.7G free.
Feb 25 13:01:48 remotehost.lab.example.com systemd[1]: Starting Tell Plymouth To
Write Out Runtime Data...
Feb 25 13:01:48 remotehost.lab.example.com systemd[1]: Started Tell Plymouth To
Write Out Runtime Data.
```

**Note**

In the **grep** above, the pipe (|) symbol acts as an or operator. That is, **grep** matches any line containing either the **Runtime** string or the **System journal** string from the **journalctl** output. This fetches the current size limits on the volatile (**Runtime**) journal store as well the persistent (**System**) journal store.

Configuring Persistent System Journals

To configure the **systemd-journald** service to preserve system journals persistently across reboot, set **Storage** to **persistent** in the **/etc/systemd/journald.conf** file. Run the text editor of your choice as the superuser to edit the **/etc/systemd/journald.conf** file.

```
[Journal]
Storage=persistent
...output omitted...
```

After editing the configuration file, restart the **systemd-journald** service to bring the configuration changes into effect.

```
[root@host ~]# systemctl restart systemd-journald
```

If the **systemd-journald** service successfully restarts, you can see that the **/var/log/journal** directory is created and contains one or more subdirectories. These subdirectories have hexadecimal characters in their long names and contain ***.journal** files. The ***.journal** files are the binary files that store the structured and indexed journal entries.

```
[root@host ~]# ls /var/log/journal
73ab164e278e48be9bf80e80714a8cd5
[root@host ~]# ls /var/log/journal/73ab164e278e48be9bf80e80714a8cd5
system.journal user-1000.journal
```

While the system journals persist across reboot, you get an extensive number of entries in the output of the **journalctl** command that includes entries from the current system boot as well as the previous ones. To limit the output to a specific system boot, use the **-b** option with the **journalctl** command. The following **journalctl** command retrieves the entries limited to the first system boot:

```
[root@host ~]# journalctl -b 1  
...output omitted...
```

The following **journalctl** command retrieves the entries limited to the second system boot. The following argument is meaningful only if the system has been rebooted at least twice:

```
[root@host ~]# journalctl -b 2
```

The following **journalctl** command retrieves the entries limited to the current system boot:

```
[root@host ~]# 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 how many prior system boots the output should include. For example, **journalctl -b -1** limits the output to only the previous boot.



References

systemd-journald.conf(5), **systemd-journald(8)** man pages

For more information refer to the *Troubleshooting problems using log files* section in the *Red Hat Enterprise Linux 8 Configuring basic system settings Guide* at https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/8/html-single/configuring_basic_system_settings/index#troubleshooting-problems-using-log-files_getting-started-with-system-administration

► Guided Exercise

Preserving the System Journal

In this exercise, you will configure the system journal to preserve its data after a reboot.

Outcomes

You should be able to configure the system journal to preserve its data after a reboot.

Before You Begin

Log in to **workstation** as **student** using **student** as the password.

On **workstation**, run **lab log-preserve start** to start the exercise. This script ensures that the environment is set up correctly.

```
[student@workstation ~]$ lab log-preserve start
```

- 1. From **workstation**, open an SSH session to **servera** as **student**.

```
[student@workstation ~]$ ssh student@servera  
...output omitted...  
[student@servera ~]$
```

- 2. As the superuser, confirm that the **/var/log/journal** directory does not exist. Use the **ls** command to list the **/var/log/journal** directory contents. Use **sudo** to elevate the **student** user privileges. Use **student** as the password if asked.

```
[student@servera ~]$ sudo ls /var/log/journal  
[sudo] password for student: student  
ls: cannot access '/var/log/journal': No such file or directory
```

As the **/var/log/journal** directory does not exist, the **systemd-journald** service is not preserving its log data.

- 3. Configure the **systemd-journald** service on **servera** to preserve journals across a reboot.

- 3.1. Uncomment the **Storage=auto** line in the **/etc/systemd/journald.conf** file and set **Storage** to **persistent**. You may use the **sudo vim /etc/systemd/journald.conf** command to edit the configuration file. Type **/ Storage=auto** from **vim** command mode to search for the **Storage=auto** line.

```
...output omitted...  
[Journal]  
Storage=persistent  
...output omitted...
```

- 3.2. Use the **systemctl** command to restart the **systemd-journald** service to bring the configuration changes into effect.

```
[student@servera ~]$ sudo systemctl restart systemd-journald.service
```

- ▶ 4. Confirm that the **systemd-journald** service on **servera** preserves its journals such that the journals persist across reboots.

- 4.1. Use the **systemctl reboot** command to restart **servera**.

```
[student@servera ~]$ sudo systemctl reboot
Connection to servera closed by remote host.
Connection to servera closed.
[student@workstation ~]$
```

Notice that the SSH connection was terminated as soon as you restarted the **servera** system.

- 4.2. Open an SSH session to **servera** again.

```
[student@workstation ~]$ ssh student@servera
...output omitted...
[student@servera ~]$
```

- 4.3. Use the **ls** command to confirm that the **/var/log/journal** directory exists. The **/var/log/journal** directory contains a subdirectory with a long hexadecimal name. The journal files are found in that directory. The subdirectory name on your system will be different.

```
[student@servera ~]$ sudo ls /var/log/journal
[sudo] password for student: student
73ab164e278e48be9bf80e80714a8cd5
[student@servera ~]$ sudo ls \
/var/log/journal/73ab164e278e48be9bf80e80714a8cd5
system.journal user-1000.journal
```

- 4.4. Log out of **servera**.

```
[student@servera ~]$ exit
logout
Connection to servera closed.
```

Finish

On **workstation**, run **lab log-preserve finish** to complete this exercise. This script ensures that the environment is restored back to the clean state.

```
[student@workstation ~]$ lab log-preserve finish
```

This concludes the guided exercise.

Maintaining Accurate Time

Objectives

After completing this section, you should be able to maintain accurate time synchronization using NTP and configure the time zone to ensure correct time stamps for events recorded by the system journal and logs.

Setting Local Clocks and Time Zones

Correct synchronized system time is critical 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.

```
[user@host ~]$ timedatectl
          Local time: Fri 2019-04-05 16:10:29 CDT
          Universal time: Fri 2019-04-05 21:10:29 UTC
                  RTC time: Fri 2019-04-05 21:10:29
                    Time zone: America/Chicago (CDT, -0500)
      System clock synchronized: yes
        NTP service: active
      RTC in local TZ: no
```

A database of time zones is available and can be listed with the **timedatectl list-timezones** command.

```
[user@host ~]$ timedatectl list-timezones
Africa/Abidjan
Africa/Accra
Africa/Addis_Ababa
Africa/Algiers
Africa/Asmara
Africa/Bamako
...
```

Time zone names are based on the public time zone database that IANA maintains. 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 change to the time zone setting of the system.

The superuser can change the system setting to update the current time zone using the **timedatectl set-timezone** command. The following **timedatectl** command updates the current time zone to **America/Phoenix**.

```
[root@host ~]# timedatectl set-timezone America/Phoenix
[root@host ~]# timedatectl
    Local time: Fri 2019-04-05 14:12:39 MST
    Universal time: Fri 2019-04-05 21:12:39 UTC
        RTC time: Fri 2019-04-05 21:12:39
        Time zone: America/Phoenix (MST, -0700)
  System clock synchronized: yes
    NTP service: active
      RTC in local TZ: no
```



Note

Should you need to use the Coordinated Universal Time (UTC) on a particular server, set its time zone to UTC. The **tzselect** command does not include the name of the UTC time zone. Use the **timedatectl set-timezone UTC** command to set the system's current time zone to **UTC**.

Use the **timedatectl set-time** command to change the system's current time. The time is specified in the "YYYY-MM-DD hh:mm:ss" format, where either date or time can be omitted. The following **timedatectl** command changes the time to **09:00:00**.

```
[root@host ~]# timedatectl set-time 9:00:00
[root@host ~]# timedatectl
    Local time: Fri 2019-04-05 09:00:27 MST
    Universal time: Fri 2019-04-05 16:00:27 UTC
        RTC time: Fri 2019-04-05 16:00:27
        Time zone: America/Phoenix (MST, -0700)
  System clock synchronized: yes
    NTP service: active
      RTC in local TZ: no
```

The **timedatectl set-ntp** command enables or disables NTP synchronization for automatic time adjustment. The option requires either a **true** or **false** argument to turn it on or off. The following **timedatectl** command turns on NTP synchronization.

```
[root@host ~]# timedatectl set-ntp true
```

**Note**

In Red Hat Enterprise Linux 8, the `timedatectl set-ntp` command will adjust whether or not **chronyd** NTP service is operating. Other Linux distributions might use this setting to adjust a different NTP or SNTP service.

Enabling or disabling NTP using other utilities in Red Hat Enterprise Linux, such as in the graphical GNOME Settings application, also updates this setting.

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. If no network connectivity is available, **chronyd** calculates the RTC clock drift, which is recorded in the **driftfile** specified in the `/etc/chrony.conf` configuration file.

By default, the **chronyd** service 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 **stratum** of the NTP time source determines its quality. 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.

The **server** and **peer** are the two categories of time sources that you can declare in the `/etc/chrony.conf` configuration file. 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.

The following **server classroom.example.com iburst** line in the `/etc/chrony.conf` file causes the **chronyd** service to use the `classroom.example.com` NTP time source.

```
# Use public servers from the pool.ntp.org project.
...output omitted...
server classroom.example.com iburst
...output omitted...
```

After pointing **chronyd** to the local time source, `classroom.example.com`, you should restart the service.

```
[root@host ~]# systemctl restart chronyd
```

The **chronyc** command acts as a client to the **chronyd** service. After setting up NTP synchronization, you should verify that the local system is seamlessly using the NTP server to synchronize the system clock using the **chronyc sources** command. For more verbose output with additional explanations about the output, use the **chronyc sources -v** command.

```
[root@host ~]# 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,
||           Log2(Polling interval) -.
||                               \           | 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 the **classroom.example.com** server has been used as a time source and is the NTP server the machine is currently synchronized to.



References

timedatectl(1), tzselect(8), chronyd(8), chrony.conf(5), and chronyc(1)
man pages

NTP Pool Project

<http://www.pool.ntp.org/>

Time Zone Database

<http://www.iana.org/time-zones>

► Guided Exercise

Maintaining Accurate Time

In this exercise, you will adjust the time zone on a server and ensure that its system clock is synchronized with an NTP time source.

Outcomes

You should be able to:

- Change the time zone on a server.
- Configure the server to synchronize its time with an NTP time source.

Before You Begin

Log in to **workstation** as **student** using **student** as the password.

On **workstation**, run **lab log-maintain start** to start the exercise. This script ensures that the time synchronization is disabled on the **servera** system to provide you with the opportunity to manually update the settings on the system and enable the time synchronization.

```
[student@workstation ~]$ lab log-maintain start
```

- 1. From **workstation**, open an SSH session to **servera** as **student**.

```
[student@workstation ~]$ ssh student@servera  
...output omitted...  
[student@servera ~]$
```

- 2. For the sake of the activity, pretend that the **servera** system is relocated to Haiti and so you need to update the time zone appropriately. Use **sudo** to elevate the privileges of the **student** user while running the **timedatectl** command to update the time zone. Use **student** as the password if asked.

- 2.1. Use the **tzselect** command to determine the appropriate time zone for Haiti.

```
[student@servera ~]$ tzselect  
Please identify a location so that time zone rules can be set correctly.  
Please select a continent, ocean, "coord", or "TZ".  
1) Africa  
2) Americas  
3) Antarctica  
4) Asia  
5) Atlantic Ocean  
6) Australia  
7) Europe  
8) Indian Ocean
```

```

9) Pacific Ocean
10) coord - I want to use geographical coordinates.
11) TZ - I want to specify the time zone using the Posix TZ format.
#? 2
Please select a country whose clocks agree with yours.
 1) Anguilla          19) Dominican Republic   37) Peru
 2) Antigua & Barbuda 20) Ecuador           38) Puerto Rico
 3) Argentina         21) El Salvador        39) St Barthelemy
 4) Aruba             22) French Guiana     40) St Kitts & Nevis
 5) Bahamas           23) Greenland         41) St Lucia
 6) Barbados          24) Grenada          42) St Maarten (Dutch)
 7) Belize            25) Guadeloupe        43) St Martin (French)
 8) Bolivia            26) Guatemala        44) St Pierre & Miquelon
 9) Brazil             27) Guyana           45) St Vincent
10) Canada            28) Haiti             46) Suriname
11) Caribbean NL      29) Honduras          47) Trinidad & Tobago
12) Cayman Islands    30) Jamaica           48) Turks & Caicos Is
13) Chile              31) Martinique        49) United States
14) Colombia          32) Mexico            50) Uruguay
15) Costa Rica         33) Montserrat       51) Venezuela
16) Cuba               34) Nicaragua         52) Virgin Islands (UK)
17) Curaçao           35) Panama           53) Virgin Islands (US)
18) Dominica          36) Paraguay
#? 28
The following information has been given:
```

Haiti

Therefore TZ='America/Port-au-Prince' will be used.
 Selected time is now: Tue Feb 19 00:51:05 EST 2019.
 Universal Time is now: Tue Feb 19 05:51:05 UTC 2019.
 Is the above information OK?

- 1) Yes
- 2) No

#? 1

You can make this change permanent for yourself by appending the line
 TZ='America/Port-au-Prince'; 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/Port-au-Prince

Notice that the preceding **tzselect** command displayed the appropriate time zone
 for Haiti.

- 2.2. Use the **timedatectl** command to update the time zone on **servera** to **America/Port-au-Prince**.

```
[student@servera ~]$ sudo timedatectl set-timezone \
America/Port-au-Prince
[sudo] password for student: student
```

- 2.3. Use the **timedatectl** command to verify that the time zone has been updated to **America/Port-au-Prince**.

```
[student@servera ~]$ timedatectl
    Local time: Tue 2019-02-19 01:16:29 EST
    Universal time: Tue 2019-02-19 06:16:29 UTC
          RTC time: Tue 2019-02-19 06:16:29
            Time zone: America/Port-au-Prince (EST, -0500)
System clock synchronized: no
          NTP service: inactive
        RTC in local TZ: no
```

- 3. Configure the **chronyd** service on **servera** to synchronize the system time with the NTP time source **classroom.example.com**.

- 3.1. Edit the **/etc/chrony.conf** file to specify the **classroom.example.com** server as the NTP time source. You may use the **sudo vim /etc/chrony.conf** command to edit the configuration file. The following output shows the configuration line you must add to the configuration file:

```
...output omitted...
server classroom.example.com iburst
...output omitted...
```

The preceding line in the **/etc/chrony.conf** configuration file includes the **iburst** option to speed up initial time synchronization.

- 3.2. Use the **timedatectl** command to turn on the time synchronization on **servera**.

```
[student@servera ~]$ sudo timedatectl set-ntp yes
```

The preceding **timedatectl** command activates the NTP server with the changed settings in the **/etc/chrony.conf** configuration file. The preceding **timedatectl** command may activate either the **chronyd** or the **ntpd** service, based on what is currently installed on the system.

- 4. Verify that the time settings on **servera** are currently configured to synchronize with the **classroom.example.com** time source in the classroom environment.

- 4.1. Use the **timedatectl** command to verify that the **servera** currently has the time synchronization enabled.

```
[student@servera ~]$ timedatectl
    Local time: Tue 2019-02-19 01:52:17 EST
    Universal time: Tue 2019-02-19 06:52:17 UTC
          RTC time: Tue 2019-02-19 06:52:17
            Time zone: America/Port-au-Prince (EST, -0500)
System clock synchronized: yes
          NTP service: active
        RTC in local TZ: no
```

**Note**

If the preceding output shows that the clock is not synchronized, wait for two seconds and re-run the **timedatectl** command. It takes a few seconds to successfully synchronize the time settings with the time source.

- 4.2. Use the **chronyc** command to verify that the **servera** system is currently synchronizing its time settings with the **classroom.example.com** time source.

```
[student@servera ~]$ 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
    ||   Reachability register (octal) -.
    ||   Log2(Polling interval) --.      |       |   xxxx = adjusted offset,
    ||                           \      |       |   yyyy = measured offset,
    ||                           |      |       |   zzzz = estimated error.
    ||
MS Name/IP address          Stratum Poll Reach LastRx Last sample
=====
^* classroom.example.com      2     6   377    62  +105us[ +143us] +/-   14ms
```

Notice that the preceding output shows an asterisk (*) in the source state (**S**) field for the **classroom.example.com** NTP time source. The asterisk indicates that the local system time is currently in successful synchronization with the NTP time source.

- 4.3. Log out of **servera**.

```
[student@servera ~]$ exit
logout
Connection to servera closed.
[student@workstation ~]$
```

Finish

On **workstation**, run **lab log-maintain finish** to complete this exercise. This script ensures that the original time zone is restored along with all the original time settings on **servera**.

```
[student@workstation ~]$ lab log-maintain finish
```

This concludes the guided exercise.

► Lab

Analyzing and Storing Logs

Performance Checklist

In this lab, you will change the time zone on an existing server and configure a new log file for all events related to authentication failures.

Outcomes

You should be able to:

- Update the time zone on an existing server.
- Configure a new log file to store all messages related to authentication failures.

Before You Begin

Log in to **workstation** as **student** using **student** as the password.

On **workstation**, run **lab log-review start** to start the exercise. This script records the current time zone of the **serverb** system and ensures that the environment is setup correctly.

```
[student@workstation ~]$ lab log-review start
```

1. From **workstation**, open an SSH session to **serverb** as **student**.
2. Pretend that the **serverb** system has been relocated to Jamaica and you must update the time zone appropriately. Use **sudo** to elevate the **student** user privileges for the **timedatectl** command to update the time zone. Use **student** as the password if asked.
3. Display the log events recorded in the previous 30 minutes on **serverb**.
4. Create the **/etc/rsyslog.d/auth-errors.conf** file, configured to have the **rsyslog** service write messages related to authentication and security issues to the new **/var/log/auth-errors** file. Use the **authpriv** facility and the **alert** priority in the configuration file.

Evaluation

On **workstation**, run the **lab log-review grade** command to confirm success of this exercise.

```
[student@workstation ~]$ lab log-review grade
```

Finish

On **workstation**, run **lab log-review finish** to complete this lab. This script ensures that the original time zone is restored along with all the original time settings on **serverb**.

```
[student@workstation ~]$ lab log-review finish
```

This concludes the guided exercise.

► Solution

Analyzing and Storing Logs

Performance Checklist

In this lab, you will change the time zone on an existing server and configure a new log file for all events related to authentication failures.

Outcomes

You should be able to:

- Update the time zone on an existing server.
- Configure a new log file to store all messages related to authentication failures.

Before You Begin

Log in to **workstation** as **student** using **student** as the password.

On **workstation**, run **lab log-review start** to start the exercise. This script records the current time zone of the **serverb** system and ensures that the environment is setup correctly.

```
[student@workstation ~]$ lab log-review start
```

1. From **workstation**, open an SSH session to **serverb** as **student**.

```
[student@workstation ~]$ ssh student@serverb
...output omitted...
[student@serverb ~]$
```

2. Pretend that the **serverb** system has been relocated to Jamaica and you must update the time zone appropriately. Use **sudo** to elevate the **student** user privileges for the **timedatectl** command to update the time zone. Use **student** as the password if asked.

- 2.1. Use the **timedatectl** command to view available time zones and determine the appropriate time zone for Jamaica.

```
[student@serverb ~]$ timedatectl list-timezones | grep America/Jamaica
America/Jamaica
```

- 2.2. Use the **timedatectl** command to set the time zone of the **serverb** system to **America/Jamaica**.

```
[student@serverb ~]$ sudo timedatectl set-timezone America/Jamaica
[sudo] password for student: student
```

- 2.3. Use the **timedatectl** command to verify that the time zone is successfully set to **America/Jamaica**.

```
[student@serverb ~]$ timedatectl
    Local time: Tue 2019-02-19 11:12:46 EST
    Universal time: Tue 2019-02-19 16:12:46 UTC
          RTC time: Tue 2019-02-19 16:12:45
        Time zone: America/Jamaica (EST, -0500)
System clock synchronized: yes
          NTP service: active
     RTC in local TZ: no
```

3. Display the log events recorded in the previous 30 minutes on **serverb**.
 - 3.1. Use the **date** command to determine the time frame to view the journal entries.

```
[student@serverb ~]$ date
Fri Feb 22 07:31:05 EST 2019
[student@serverb ~]$ date -d "-30 minutes"
Fri Feb 22 07:01:31 EST 2019
```
 - 3.2. Use the **journalctl** command **--since** and **--until** options to display log events recorded in the previous 30 minutes on **serverb**. To quit **journalctl**, press **q**.

```
[student@serverb ~]$ journalctl --since 07:01:00 --until 07:31:00
...output omitted...
Feb 22 07:24:28 serverb.lab.example.com systemd[1138]: Reached target Timers.
Feb 22 07:24:28 serverb.lab.example.com systemd[1138]: Reached target Paths.
Feb 22 07:24:28 serverb.lab.example.com systemd[1138]: Starting D-Bus User Message Bus Socket.
Feb 22 07:24:28 serverb.lab.example.com systemd[1138]: Listening on D-Bus User Message Bus Socket.
Feb 22 07:24:28 serverb.lab.example.com systemd[1138]: Reached target Sockets.
Feb 22 07:24:28 serverb.lab.example.com systemd[1138]: Reached target Basic System.
Feb 22 07:24:28 serverb.lab.example.com systemd[1138]: Reached target Default.
Feb 22 07:24:28 serverb.lab.example.com systemd[1138]: Startup finished in 123ms.
Feb 22 07:24:28 serverb.lab.example.com systemd[1]: Started User Manager for UID 1000.
Feb 22 07:24:28 serverb.lab.example.com sshd[1134]: pam_unix(sshd:session): session opened for user student by (uid=0)
Feb 22 07:26:56 serverb.lab.example.com systemd[1138]: Starting Mark boot as successful...
Feb 22 07:26:56 serverb.lab.example.com systemd[1138]: Started Mark boot as successful.
lines 1-36/36 (END) q
[student@serverb ~]$
```
4. Create the **/etc/rsyslog.d/auth-errors.conf** file, configured to have the **rsyslog** service write messages related to authentication and security issues to the new **/var/log/auth-errors** file. Use the **authpriv** facility and the **alert** priority in the configuration file.
 - 4.1. Create the **/etc/rsyslog.d/auth-errors.conf** file to specify the new **/var/log/auth-errors** file as the destination for messages related to authentication and security issues. You may use the **sudo vim /etc/rsyslog.d/auth-errors.conf** command to create the configuration file.

```
authpriv.alert /var/log/auth-errors
```

- 4.2. Restart the **rsyslog** service so that the changes in the configuration file take effect.

```
[student@serverb ~]$ sudo systemctl restart rsyslog
```

- 4.3. Use the **logger** command to write a new log message to the **/var/log/auth-errors** file. Apply the **-p authpriv.alert** option to generate a log message relevant to authentication and security issues.

```
[student@serverb ~]$ logger -p authpriv.alert "Logging test authpriv.alert"
```

- 4.4. Use the **tail** command to confirm that the **/var/log/auth-errors** file contains the log entry with the **Logging test authpriv.alert** message.

```
[student@serverb ~]$ sudo tail /var/log/auth-errors
Feb 19 11:56:07 serverb student[6038]: Logging test authpriv.alert
```

- 4.5. Log out of **serverb**.

```
[student@serverb ~]$ exit
logout
Connection to serverb closed.
[student@workstation ~]$
```

Evaluation

On **workstation**, run the **lab log-review grade** command to confirm success of this exercise.

```
[student@workstation ~]$ lab log-review grade
```

Finish

On **workstation**, run **lab log-review finish** to complete this lab. This script ensures that the original time zone is restored along with all the original time settings on **serverb**.

```
[student@workstation ~]$ lab log-review finish
```

This concludes the guided exercise.

Summary

In this chapter, you learned:

- The **systemd-journald** and **rsyslog** services capture and write log messages to the appropriate files.
- The **/var/log** directory contains log files.
- Periodic rotation of log files prevent them from filling up the file system space.
- The **systemd** journals are temporary and do not persist across reboot.
- The **chrony** service helps to synchronize time settings with a time source.
- The time zone of the server can be updated based on its location.

Chapter 12

Managing Networking

Goal

Configure network interfaces and settings on Red Hat Enterprise Linux servers.

Objectives

- Describe fundamental concepts of network addressing and routing for a server.
- Test and inspect current network configuration with command-line utilities.
- Manage network settings and devices using `nmcli`.
- Modify network settings by editing configuration files.
- Configure a server's static host name and its name resolution, and test the results.

Sections

- Describing Networking Concepts (and Quiz)
- Validating Network Configuration (and Guided Exercise)
- Configuring Networking from the Command Line (and Guided Exercise)
- Editing Network Configuration Files (and Guided Exercise)
- Configuring Host Names and Name Resolution (and Guided Exercise)

Lab

Managing Networking

Describing Networking Concepts

Objectives

After completing this section, you should be able to describe fundamental concepts of network addressing and routing for a server.

TCP/IP Network Model

The *TCP/IP network model* is a simplified, four-layered set of abstractions that describes how different protocols interoperate in order for computers to send traffic from one machine to another over the Internet. It is specified by RFC 1122, *Requirements for Internet Hosts -- Communication Layers*. The four layers are:

- **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. The IPv4 and IPv6 protocols are Internet layer protocols. Each host has an IP address and a prefix used to determine network addresses. Routers are used to connect networks.

- **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.

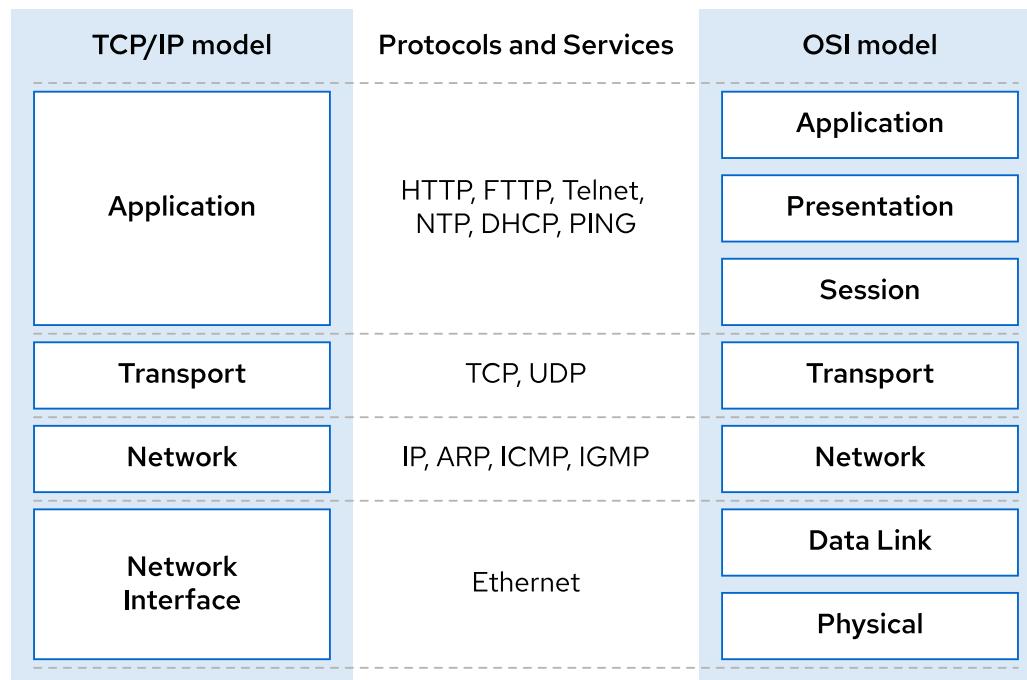


Figure 12.1: Comparison of the TCP/IP and OSI network models

Describing Network Interface Names

Each network port on a system has a name, which you use to configure and identify it.

Older versions of Red Hat Enterprise Linux used names like **eth0**, **eth1**, and **eth2** for each network interface. The name **eth0** was the first network port detected by the operating system, **eth1** the second, and so on. However, as devices are added and removed, the mechanism detecting devices and naming them could change which interface gets which name. Furthermore, the PCIe standard does not guarantee the order in which PCIe devices will be detected on boot, which could change device naming unexpectedly due to variations during device or system startup.

Newer versions of Red Hat Enterprise Linux use a different naming system. Instead of being based on detection order, the names of network interfaces are assigned based on information from the firmware, the PCI bus topology, and type of network device.

Network interface names start with the type of interface:

- Ethernet interfaces begin with **en**
- WLAN interfaces begin with **wl**
- WWAN interfaces begin with **ww**

The rest of the interface name after the type will be based on information provided by the server's firmware or determined by the location of the device in the PCI topology.

- **oN** indicates that this is an on-board device and the server's firmware provided index number *N* for the device. So **eno1** is on-board Ethernet device 1. Many servers will not provide this information.
- **sN** indicates that this device is in PCI hotplug slot *N*. So **ens3** is an Ethernet card in PCI hotplug slot 3.

- **pMsN** indicates that this is a PCI device on bus M in slot N. So **wlp4s0** is a WLAN card on PCI bus 4 in slot 0. If the card is a multi-function device (possible with an Ethernet card with multiple ports, or devices that have Ethernet plus some other functionality), you may see **fN** added to the device name. So **enp0s1f0** is function 0 of the Ethernet card on bus 0 in slot 1. There might also be a second interface named **enp0s1f1** that is function 1 of that same device.

Persistent naming means that once you know what the name is for a network interface on the system, you also know that it will not change later. The trade off is that you cannot assume that a system with one interface will name that interface **eth0**.

IPv4 Networking

IPv4 is the primary network protocol used on the Internet today. You should have at least a basic understanding of IPv4 networking in order to manage network communication for your servers.

IPv4 Addresses

An IPv4 address is a 32-bit number, normally expressed in decimal as four 8-bit 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 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 uses 24 bits for the network part and reads **255.255.255.0**. A newer syntax, called CIDR notation, specifies 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 following examples illustrate how the IP address, prefix (netmask), network part, and host part are related.

IP Address:

172.17.5.3 = **10101100.00010001.00000101.00000011**

Netmask:

255.255.0.0 = **11111111.11111111.00000000.00000000**

**IP Address:**

192.168.5.3 = **11000000.10101000.00000101.00000011**

Netmask:

255.255.255.0 = **11111111.11111111.11111111.00000000**



Figure 12.2: IPv4 addresses and netmasks

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.

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. A routing table entry lists a destination network, which interface to use when sending traffic, and the IP address of any intermediate router required to relay a message to its final destination. The routing table entry matching 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.

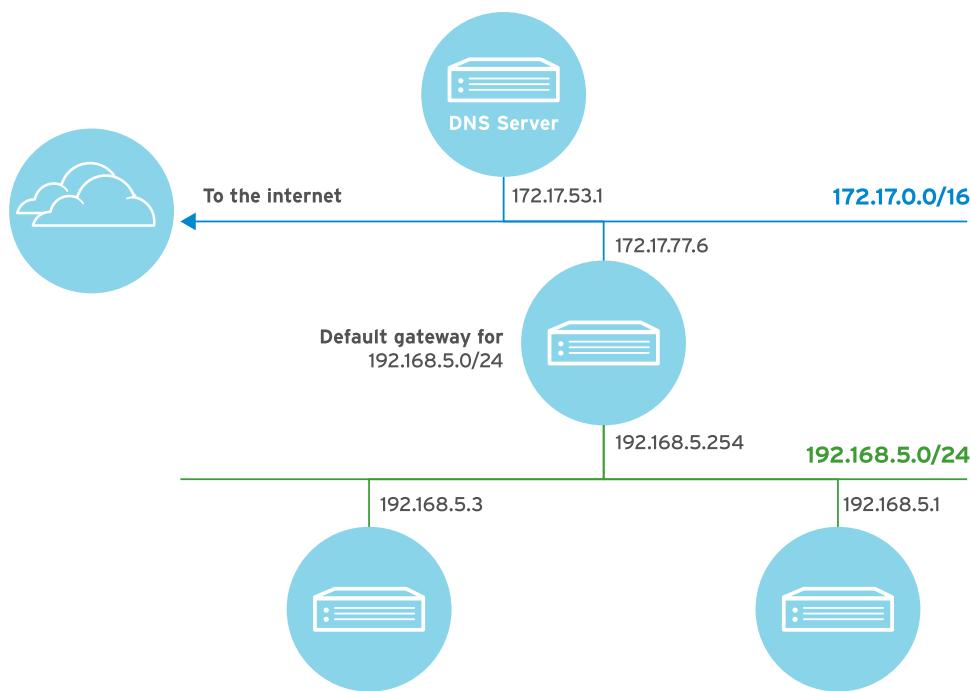


Figure 12.3: Example network topology

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 (default)	enp3s0	192.168.5.254

In this example, traffic headed for the IP address **192.0.2.102** from this host is 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** is 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** is transmitted out the **enp3s0** Ethernet interface to a router at **192.168.5.254**, which forwards 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 uses its own routing table to determine where to forward that traffic to next.

IPv4 Address and Route Configuration

A server can automatically configure its IPv4 network settings at boot time from a *DHCP* server. A local client daemon queries the link for a server and network settings, and obtains a *lease* to use those settings for a specific length of time. If the client does not request a renewal of the lease periodically, it might lose its network configuration settings.

As an alternative, you can configure a server to use a *static* network configuration. In this case, network settings are read from local configuration files. You must get the correct settings from your network administrator and update them manually as needed to avoid conflicts with other servers.

IPv6 Networking

IPv6 is intended as an eventual replacement for the IPv4 network protocol. You will need to understand how it works since increasing numbers of production systems use IPv6 addressing. For example, many ISPs already use IPv6 for internal communication and device management networks in order to preserve scarce IPv4 addresses for customer purposes.

IPv6 can also be used in parallel with IPv4 in a *dual-stack* model. In this configuration, a network interface can have an IPv6 address or addresses as well as IPv4 addresses. Red Hat Enterprise Linux operates in a dual-stack mode by default.

IPv6 Addresses

An IPv6 address is a 128-bit number, normally expressed as eight colon-separated groups of four hexadecimal nibbles (half-bytes). Each nibble represents four bits of the IPv6 address, so each group represents 16 bits of the IPv6 address.

```
2001:0db8:0000:0010:0000:0000:0000:0001
```

To make IPv6 addresses easier to write, leading zeros in a colon-separated group do not need to be written. However, at least one hexadecimal digit must be written in each colon-separated group.

2001:db8:0:10:0:0:0:1

Since addresses with long strings of zeros are common, one or more consecutive groups of zeros only may be combined with *exactly one :: block*.

2001:db8:0:10::1

Notice that under these rules, **2001:db8::0010:0:0:0:1** would be another less convenient way to write the example address. But it is a valid representation of the same address, and this can confuse administrators new to IPv6. Some tips for writing consistently readable addresses:

- Suppress leading zeros in a group.
- Use :: to shorten as much as possible.
- If an address contains two consecutive groups of zeros, equal in length, it is preferred to shorten the leftmost groups of zeros to :: and the rightmost groups to :0: for each group.
- Although it is allowed, do not use :: to shorten one group of zeros. Use :0: instead, and save :: for consecutive groups of zeros.
- Always use lowercase letters for hexadecimal numbers **a** through **f**.

**Important**

When including a TCP or UDP network port after an IPv6 address, always enclose the IPv6 address in square brackets so that the port does not look like it is part of the address.

[2001:db8:0:10::1]:80

IPv6 Subnetting

A normal IPv6 unicast address is divided into two parts: the *network prefix* and *interface ID*. The network prefix identifies the subnet. No two network interfaces on the same subnet can have the same interface ID; the interface ID identifies a particular interface on the subnet.

Unlike IPv4, IPv6 has a standard subnet mask, which is used for almost all normal addresses, /64. In this case, half of the address is the network prefix and half of it is the interface ID. This means that a single subnet can hold as many hosts as necessary.

Typically, the network provider will *allocate* a shorter prefix to an organization, such as a /48. This leaves the rest of the network part for assigning subnets (always of length /64) from that allocated prefix. For a /48 allocation, that leaves 16 bits for subnets (up to 65536 subnets).

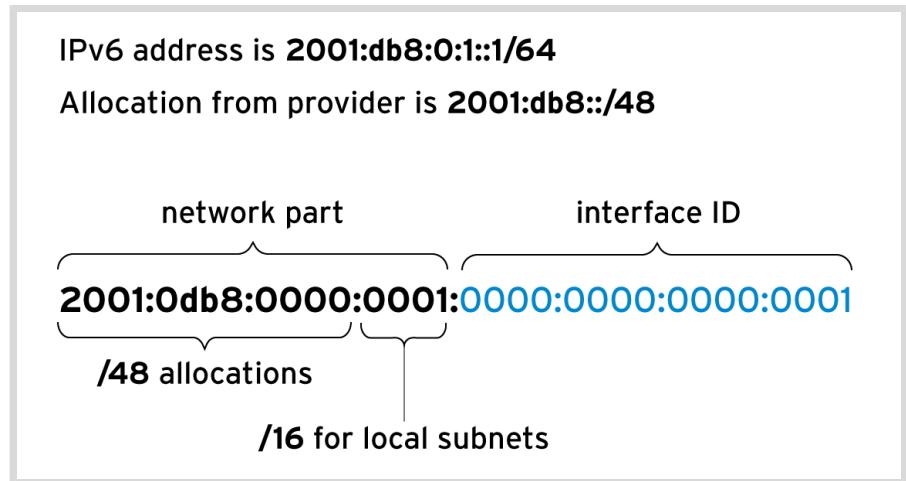


Figure 12.4: IPv6 address parts and subnetting

Common IPv6 Addresses and Networks

IPv6 address or network	Purpose	Description
::1/128	localhost	The IPv6 equivalent to 127.0.0.1/8 , set on the loopback interface.
::	The unspecified address	The IPv6 equivalent to 0.0.0.0 . For a network service, this could indicate that it is listening on all configured IP addresses.
::/0	The default route (the IPv6 Internet)	The IPv6 equivalent to 0.0.0.0/0 . The default route in the routing table matches this network; the router for this network is where all traffic, for which there is no better route, is sent.
2000::/3	Global unicast addresses	"Normal" IPv6 addresses are currently being allocated from this space by IANA. This is equivalent to all the networks ranging from 2000::/16 through 3fff::/16 .
fd00::/8	Unique local addresses (RFC 4193)	IPv6 has no direct equivalent of RFC 1918 private address space, although this is close. A site can use these to self-allocate a private routable IP address space inside the organization, but these networks cannot be used on the global Internet. The site must <i>randomly</i> select a /48 from this space, but it can subnet the allocation into /64 networks normally.

IPv6 address or network	Purpose	Description
fe80::/10	Link-local addresses	Every IPv6 interface automatically configures a <i>link-local</i> unicast address that only works on the local link on the fe80::/64 network. However, the entire fe80::/10 range is reserved for future use by the local link. This will be discussed in more detail later.
ff00::/8	Multicast	The IPv6 equivalent to 224.0.0.0/4 . Multicast is used to transmit to multiple hosts at the same time, and is particularly important in IPv6 because it has no broadcast addresses.

**Important**

The table above lists network address *allocations* that are reserved for specific purposes. These allocations may consist of many different networks. Remember that IPv6 networks allocated from the global unicast and link-local unicast spaces have a standard **/64** subnet mask.

A *link-local address* in IPv6 is an unroutable address used only to talk to hosts on a specific network link. Every network interface on the system is automatically configured with a link-local address on the **fe80::/64** network. To ensure that it is unique, the interface ID of the link-local address is constructed from the network interface's Ethernet hardware address. The usual procedure to convert the 48-bit MAC address to a 64-bit interface ID is to invert bit 7 of the MAC address and insert **ff:fe** between its two middle bytes.

- Network prefix: **fe80::/64**
- MAC address: **00:11:22:aa:bb:cc**
- Link-local address: **fe80::211:22ff:fea:bbcc/64**

The link-local addresses of other machines can be used like normal addresses by other hosts on the same link. Since every link has a **fe80::/64** network on it, the routing table cannot be used to select the outbound interface correctly. The link to use when talking to a link-local address must be specified with a *scope identifier* at the end of the address. The scope identifier consists of % followed by the name of the network interface.

For example, to use **ping6** to ping the link-local address **fe80::211:22ff:fea:bbcc** using the link connected to the **ens3** network interface, the correct command syntax is the following:

```
[user@host ~]$ ping6 fe80::211:22ff:fea:bbcc%ens3
```

**Note**

Scope identifiers are only needed when contacting addresses that have “link” scope. Normal global addresses are used just like they are in IPv4, and select their outbound interfaces from the routing table.

Multicast allows one system to send traffic to a special IP address that is received by multiple systems. It differs from broadcast since only specific systems on the network receive the traffic. It also differs from broadcast in IPv4 since some multicast traffic might be routed to other subnets, depending on the configuration of your network routers and systems.

Multicast plays a larger role in IPv6 than in IPv4 because there is no broadcast address in IPv6. One key multicast address in IPv6 is **ff02::1**, the **all-nodes** link-local address. Pinging this address sends traffic to all nodes on the link. Link-scope multicast addresses (starting **ff02::/8**) need to be specified with a scope identifier, just like a link-local address.

```
[user@host ~]$ ping6 ff02::1%ens3
PING ff02::1%ens3(ff02::1) 56 data bytes
64 bytes from fe80::211:22ff:feaa:bbcc: icmp_seq=1 ttl=64 time=0.072 ms
64 bytes from fe80::200:aaff:fe33:2211: icmp_seq=1 ttl=64 time=102 ms (DUP!)
64 bytes from fe80::bcd:efff:fea1:b2c3: icmp_seq=1 ttl=64 time=103 ms (DUP!)
64 bytes from fe80::211:22ff:feaa:bbcc: icmp_seq=2 ttl=64 time=0.079 ms
...output omitted...
```

IPv6 Address Configuration

IPv4 has two ways in which addresses get configured on network interfaces. Network addresses may be configured on interfaces manually by the administrator, or dynamically from the network using DHCP. IPv6 also supports manual configuration, and two methods of dynamic configuration, one of which is DHCPv6.

Interface IDs for static IPv6 addresses can be selected at will, just like IPv4. In IPv4, there are two addresses on a network that could not be used: the lowest address in the subnet and the highest address in the subnet. In IPv6, the following interface IDs are reserved and cannot be used for a normal network address on a host.

- The all-zeros identifier **0000:0000:0000:0000** (“subnet router anycast”) used by all routers on the link. (For the **2001:db8::/64** network, this would be the address **2001:db8::**)
- The identifiers **fdff:ffff:ffff:ff80** through **fdff:ffff:ffff:ffff**.

DHCPv6 works differently than DHCP for IPv4, because there is no broadcast address. Essentially, a host sends a DHCPv6 request from its link-local address to port 547/UDP on **ff02::1:2**, the **all-dhcp-servers** link-local multicast group. The DHCPv6 server then usually sends a reply with appropriate information to port 546/UDP on the client's link-local address.

The *dhcp* package in Red Hat Enterprise Linux 8 provides support for a DHCPv6 server.

In addition to DHCPv6, IPv6 also supports a second dynamic configuration method, called *Stateless Address Autoconfiguration (SLAAC)*. Using SLAAC, the host brings up its interface with a link-local **fe80::/64** address normally. It then sends a “router solicitation” to **ff02::2**, the all-routers link-local multicast group. An IPv6 router on the local link responds to the host's link-local address with a network prefix and possibly other information. The host then uses that network prefix with an interface ID that it normally constructs in the same way that link-local addresses are constructed. The router periodically sends multicast updates (“router advertisements”) to confirm or update the information it provided.

The *radvd* package in Red Hat Enterprise Linux 8 allows a Red Hat Enterprise Linux based IPv6 router to provide SLAAC through router advertisements.

**Important**

A typical Red Hat Enterprise Linux 8 machine configured to get IPv4 addresses through DHCP is usually also configured to use SLAAC to get IPv6 addresses. This can result in machines unexpectedly obtaining IPv6 addresses when an IPv6 router is added to the network.

Some IPv6 deployments combine SLAAC and DHCPv6, using SLAAC to only provide network address information and DHCPv6 to provide other information, such as which DNS servers and search domains to configure.

Host Names and IP Addresses

It would be inconvenient if you always had to use IP addresses to contact your servers. Humans generally would prefer to work with names than long and hard-to-remember strings of numbers. And so Linux has a number of mechanisms to map a host name to an IP address, collectively called *name resolution*.

One way is to set a static entry for each name in the **/etc/hosts** file on each system. This requires you to manually update each server's copy of the file.

For most hosts, you can look up the address for a host name (or a host name from an address) from a network service called the *Domain Name System (DNS)*. DNS is a distributed network of servers providing mappings of host names to IP addresses. In order for name service to work, a 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. This is typically configured through DHCP or a static setting in a file called **/etc/resolv.conf**. Later sections of this chapter will discuss how to configure name resolution.



References

services(5), ping(8), biosdevname(1), and udev(7) man pages

For more information, refer to the *Configuring and Managing Networking Guide* at
https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/8/html-single/configuring_and_managing_networking/

Understanding systemd's predictable network device names

<https://major.io/2015/08/21/understanding-systemds-predictable-network-device-names/>

Selected IETF RFC references:

RFC 2460: Internet Protocol, Version 6 (IPv6) Specification

<http://tools.ietf.org/html/rfc2460>

RFC 4291: IP Version 6 Addressing Architecture

<http://tools.ietf.org/html/rfc4291>

RFC 5952: A Recommendation For IPv6 Address Text Representation

<http://tools.ietf.org/html/rfc5952>

RFC 4862: IPv6 Stateless Address Autoconfiguration

<http://tools.ietf.org/html/rfc4862>

RFC 3315: Dynamic Host Configuration Protocol for IPv6 (DHCPv6)

<http://tools.ietf.org/html/rfc3315>

RFC 3736: Stateless Dynamic Host Configuration Protocol (DHCP) Service for IPv6

<http://tools.ietf.org/html/rfc3736>

RFC 4193: Unique Local IPv6 Unicast Addresses

<http://tools.ietf.org/html/rfc4193>

► Quiz

Describing Networking Concepts

Choose the correct answer to the following questions:

► 1. Which number is the size, in bits, of an IPv4 address?

- a. 4
- b. 8
- c. 16
- d. 32
- e. 64
- f. 128

► 2. Which term determines how many leading bits in the IP address contribute to its network address?

- a. netscope
- b. netmask
- c. subnet
- d. multicast
- e. netaddr
- f. network

► 3. Which address represents a valid IPv4 host address on the 192.168.1.0/24 network?

- a. 192.168.1.188
- b. 192.168.1.0
- c. 192.168.1.255
- d. 192.168.1.256

► 4. Which number is the size, in bits, of an IPv6 address?

- a. 4
- b. 8
- c. 16
- d. 32
- e. 64
- f. 128

► **5. Which address does not represent a valid IPv6 address?**

- a. 2000:0000:0000:0000:0000:0000:0000:0001
- b. 2::1
- c. ::
- d. ff02::1:0:0
- e. 2001:3::7:0:2
- f. 2001:db8::7::2
- g. 2000::1

► **6. Which term allows one system to send traffic to a special IP address that is received by multiple systems?**

- a. netscope
- b. netmask
- c. subnet
- d. multicast
- e. netaddr
- f. network

► Solution

Describing Networking Concepts

Choose the correct answer to the following questions:

► 1. Which number is the size, in bits, of an IPv4 address?

- a. 4
- b. 8
- c. 16
- d. 32
- e. 64
- f. 128

► 2. Which term determines how many leading bits in the IP address contribute to its network address?

- a. netscope
- b. netmask
- c. subnet
- d. multicast
- e. netaddr
- f. network

► 3. Which address represents a valid IPv4 host address on the 192.168.1.0/24 network?

- a. 192.168.1.188
- b. 192.168.1.0
- c. 192.168.1.255
- d. 192.168.1.256

► 4. Which number is the size, in bits, of an IPv6 address?

- a. 4
- b. 8
- c. 16
- d. 32
- e. 64
- f. 128

► **5. Which address does not represent a valid IPv6 address?**

- a. 2000:0000:0000:0000:0000:0000:0000:0001
- b. 2::1
- c. ::
- d. ff02::1:0:0
- e. 2001:3::7:0:2
- f. 2001:db8::7::2
- g. 2000::1

► **6. Which term allows one system to send traffic to a special IP address that is received by multiple systems?**

- a. netscope
- b. netmask
- c. subnet
- d. multicast
- e. netaddr
- f. network

Validating Network Configuration

Objectives

After completing this section, you should be able to test and inspect current network configuration with command-line utilities.

Gathering Network Interface Information

Identifying Network Interfaces

The `ip link` command will list all network interfaces available on your system:

```
[user@host ~]$ ip link show
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN mode DEFAULT
    group default qlen 1000
        link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
2: ens3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP mode DEFAULT
    group default qlen 1000
        link/ether 52:54:00:00:00:0a brd ff:ff:ff:ff:ff:ff
3: ens4: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP mode DEFAULT
    group default qlen 1000
        link/ether 52:54:00:00:00:1e brd ff:ff:ff:ff:ff:ff
```

In the preceding example, the server has three network interfaces: `lo`, which is the loopback device that is connected to the server itself, and two Ethernet interfaces, `ens3` and `ens4`.

To configure each network interface correctly, you need to know which one is connected to which network. In many cases, you will know the MAC address of the interface connected to each network, either because it is physically printed on the card or server, or because it is a virtual machine and you know how it is configured. The MAC address of the device is listed after `link/ether` for each interface. So you know that the network card with the MAC address `52:54:00:00:00:0a` is the network interface `ens3`.

Displaying IP Addresses

Use the `ip` command to view device and address information. A single network interface can have multiple IPv4 or IPv6 addresses.

```
[user@host ~]$ ip addr show ens3
2: ens3: <BROADCAST,MULTICAST,①UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP
    qlen 1000
        ②link/ether 52:54:00:00:00:0b brd ff:ff:ff:ff:ff:ff
        ③inet 192.0.2.2/24 brd 192.0.2.255 scope global ens3
            valid_lft forever preferred_lft forever
        ④inet6 2001:db8:0:1:5054:ff:fe00:b/64 scope global
            valid_lft forever preferred_lft forever
        ⑤inet6 fe80::5054:ff:fe00:b/64 scope link
            valid_lft forever preferred_lft forever
```

- ➊ An active interface is **UP**.
- ➋ The **link/ether** line specifies the hardware (MAC) address of the device.
- ➌ The **inet** line shows an IPv4 address, its network prefix length, and scope.
- ➍ The **inet6** line shows an IPv6 address, its network prefix length, and scope. This address is of *global* scope and is normally used.
- ➎ This **inet6** line shows that the interface has an IPv6 address of *link* scope that can only be used for communication on the local Ethernet link.

Displaying Performance Statistics

The **ip** command may also be used to show statistics about network performance. Counters for each network interface can be used to identify the presence of network issues. The counters record statistics for things like the number of received (RX) and transmitted (TX) packets, packet errors, and packets that were dropped.

```
[user@host ~]$ ip -s link show ens3
2: ens3: <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       0
        TX: bytes   packets   errors   dropped carrier collsns
            300556      3250       0       0       0       0
```

Checking Connectivity Between Hosts

The **ping** command is used to test connectivity. The command continues to run until **Ctrl+c** is pressed unless options are given to limit the number of packets sent.

```
[user@host ~]$ ping -c3 192.0.2.254
PING 192.0.2.1 (192.0.2.254) 56(84) bytes of data.
64 bytes from 192.0.2.254: icmp_seq=1 ttl=64 time=4.33 ms
64 bytes from 192.0.2.254: icmp_seq=2 ttl=64 time=3.48 ms
64 bytes from 192.0.2.254: icmp_seq=3 ttl=64 time=6.83 ms

--- 192.0.2.254 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2003ms
rtt min/avg/max/mdev = 3.485/4.885/6.837/1.424 ms
```

The **ping6** command is the IPv6 version of **ping** in Red Hat Enterprise Linux. It communicates over IPv6 and takes IPv6 addresses, but otherwise works like **ping**.

```
[user@host ~]$ ping6 2001:db8:0:1::1
PING 2001:db8:0:1::1(2001:db8:0:1::1) 56 data bytes
64 bytes from 2001:db8:0:1::1: icmp_seq=1 ttl=64 time=18.4 ms
64 bytes from 2001:db8:0:1::1: icmp_seq=2 ttl=64 time=0.178 ms
64 bytes from 2001:db8:0:1::1: icmp_seq=3 ttl=64 time=0.180 ms
^C
--- 2001:db8:0:1::1 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2001ms
rtt min/avg/max/mdev = 0.178/6.272/18.458/8.616 ms
[user@host ~]$
```

When you ping link-local addresses and the link-local all-nodes multicast group (**ff02::1**), the network interface to use must be specified explicitly with a scope zone identifier (such as **ff02::1%ens3**). If this is left out, the error *connect: Invalid argument* is displayed.

Pinging **ff02::1** can be useful for finding other IPv6 nodes on the local network.

```
[user@host ~]$ ping6 ff02::1%ens4
PING ff02::1%ens4(ffff:fed2:f97b) 56 data bytes
64 bytes from fe80::78cf:7fff:fed2:f97b: icmp_seq=1 ttl=64 time=22.7 ms
64 bytes from fe80::f482:dbff:fe25:6a9f: icmp_seq=1 ttl=64 time=30.1 ms (DUP!)
64 bytes from fe80::78cf:7fff:fed2:f97b: icmp_seq=2 ttl=64 time=0.183 ms
64 bytes from fe80::f482:dbff:fe25:6a9f: icmp_seq=2 ttl=64 time=0.231 ms (DUP!)
^C
--- ff02::1%ens4 ping statistics ---
2 packets transmitted, 2 received, +2 duplicates, 0% packet loss, time 1001ms
rtt min/avg/max/mdev = 0.183/13.320/30.158/13.374 ms
[user@host ~]$ ping6 -c 1 fe80::f482:dbff:fe25:6a9f%ens4
PING fe80::f482:dbff:fe25:6a9f%ens4(fe80::f482:dbff:fe25:6a9f) 56 data bytes
64 bytes from fe80::f482:dbff:fe25:6a9f: icmp_seq=1 ttl=64 time=22.9 ms

--- fe80::f482:dbff:fe25:6a9f%ens4 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 22.903/22.903/22.903/0.000 ms
```

Remember that IPv6 link-local addresses can be used by other hosts on the same link, just like normal addresses.

```
[user@host ~]$ ssh fe80::f482:dbff:fe25:6a9f%ens4
user@fe80::f482:dbff:fe25:6a9f%ens4's password:
Last login: Thu Jun  5 15:20:10 2014 from host.example.com
[user@server ~]$
```

Troubleshooting Routing

Network routing is complex, and sometimes traffic does not behave as you might have expected. Here are some useful diagnosis tools.

Displaying the Routing Table

Use the **ip** command with the **route** option to show routing information.

```
[user@host ~]$ ip route
default via 192.0.2.254 dev ens3 proto static metric 1024
192.0.2.0/24 dev ens3 proto kernel scope link src 192.0.2.2
10.0.0.0/8 dev ens4 proto kernel scope link src 10.0.0.11
```

This shows the IPv4 routing table. All packets destined for the **10.0.0.0/8** network are sent directly to the destination through the device **ens4**. All packets destined for the **192.0.2.0/24** network are sent directly to the destination through the device **ens3**. All other packets are sent to the default router located at **192.0.2.254**, and also through device **ens3**.

Add the **-6** option to show the IPv6 routing table:

```
[user@host ~]$ ip -6 route
unreachable ::/96 dev lo metric 1024 error -101
unreachable ::ffff:0.0.0.0/96 dev lo metric 1024 error -101
2001:db8:0:1::/64 dev ens3 proto kernel metric 256
unreachable 2002:a00::/24 dev lo metric 1024 error -101
unreachable 2002:7f00::/24 dev lo metric 1024 error -101
unreachable 2002:a9fe::/32 dev lo metric 1024 error -101
unreachable 2002:ac10::/28 dev lo metric 1024 error -101
unreachable 2002:c0a8::/32 dev lo metric 1024 error -101
unreachable 2002:e000::/19 dev lo metric 1024 error -101
unreachable 3ffe:ffff::/32 dev lo metric 1024 error -101
fe80::/64 dev ens3 proto kernel metric 256
default via 2001:db8:0:1::ffff dev ens3 proto static metric 1024
```

In this example, ignore the unreachable routes, which point at unused networks. That leaves three routes:

1. The **2001:db8:0:1::/64** network, using the **ens3** interface (which presumably has an address on that network).
2. The **fe80::/64** network, using the **ens3** interface, for the link-local address. On a system with multiple interfaces, there is a route to **fe80::/64** out each interface for each link-local address.
3. A default route to all networks on the IPv6 Internet (the **::/0** network) that do not have a more specific route on the system, through the router at **2001:db8:0:1::ffff**, reachable with the **ens3** device.

Tracing Routes Taken by Traffic

To trace the path that network traffic takes to reach a remote host through multiple routers, use either **traceroute** or **tracepath**. This can identify whether an issue is with one of your routers or an intermediate one. Both commands use UDP packets to trace a path by default; 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. Typically, however, the **traceroute** command is not installed by default.

```
[user@host ~]$ tracepath access.redhat.com
...output omitted...
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.atlqamq46w-bcr00.tbone.rr.com 107.286ms asymm 12
...output omitted...
```

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. The **asymm** indication means traffic reached that router and returned from that router using different (*asymmetric*) routes. The routers shown are the ones used for outbound traffic, not the return traffic.

The **tracepath6** and **traceroute -6** commands are the equivalent to **tracepath** and **traceroute** for IPv6.

```
[user@host ~]$ tracepath6 2001:db8:0:2::451
1?: [LOCALHOST]                                0.091ms pmtu 1500
1:  2001:db8:0:1::ba                          0.214ms
2:  2001:db8:0:1::1                           0.512ms
3:  2001:db8:0:2::451                         0.559ms reached
Resume: pmtu 1500 hops 3 back 3
```

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. The **ss** command is meant to replace the older tool **netstat**, part of the **net-tools** package, which may be more familiar to some system administrators but which is not always installed.

```
[user@host ~]$ ss -ta
State      Recv-Q Send-Q      Local Address:Port          Peer Address:Port
LISTEN     0       128           *:sunrpc                  *:*
LISTEN     0       128           ①*:ssh                   *:*
LISTEN     0       100          ②127.0.0.1:smtp          *:*
LISTEN     0       128           *:36889                  *:*
ESTAB      0       0             ③172.25.250.10:ssh      172.25.254.254:59392
LISTEN     0       128           :::sunrpc                 :::*
LISTEN     0       128           ④:::ssh                  :::*
LISTEN     0       100          ⑤::1:smtp                :::*
LISTEN     0       128           :::34946                  :::*
```

- ① The port used for SSH is listening on all IPv4 addresses. The “*” is used to represent “all” when referencing IPv4 addresses or ports.
- ② The port used for SMTP is listening on the **127.0.0.1** IPv4 loopback interface.
- ③ The established SSH connection is on the 172.25.250.10 interface and originates from a system with an address of **172.25.254.254**.
- ④ The port used for SSH is listening on all IPv6 addresses. The “::” syntax is used to represent all IPv6 interfaces.
- ⑤ 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.

Option	Description
-p	Show the process using the sockets.
-A inet	Display active connections (but not listening sockets) for the inet address family. That is, ignore local UNIX domain sockets. For ss , both IPv4 and IPv6 connections are displayed. For netstat , only IPv4 connections are displayed. (netstat -A inet6 displays IPv6 connections, and netstat -46 displays IPv4 and IPv6 at the same time.)



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

For more information, refer to the *Configuring and Managing Networking Guide* at
https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/8/html-single/configuring_and_managing_networking/

► Guided Exercise

Validating Network Configuration

In this exercise, you will inspect the network configuration of one of your servers.

Outcomes

Identify the current network interfaces and basic network addresses.

Before You Begin

Log in as the **student** user on **workstation** using **student** as the password.

From **workstation**, run the **lab net-validate start** command. The command runs a start script that determine if the host, **servera**, is reachable on the network.

```
[student@workstation ~]$ lab net-validate start
```

- ▶ 1. Use the **ssh** command to log in to **servera** as the **student** user. The systems are configured to use SSH keys for authentication and passwordless access to **servera**.

```
[student@workstation ~]$ ssh student@servera
...output omitted...
[student@servera ~]$
```



Important

Network interface names are determined by their bus type and the detection order of devices during boot. Your network interface names will vary according to the course platform and hardware in use.

On your system now, locate the interface name (such as **ens06** or **en1p2**) associated with the Ethernet address **52:54:00:00:fa:0a**. Use this interface name to replace the **enX** placeholder used throughout this exercise.

Locate the network interface name associated with the Ethernet address **52:54:00:00:fa:0a**. Record or remember this name and use it to replace the **enX** placeholder in subsequent commands.

```
[student@servera ~]$ ip link
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default
    qlen 1000
        link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
2: enX: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group
    default qlen 1000
        link/ether 52:54:00:00:fa:0a brd ff:ff:ff:ff:ff:ff
```

- 3. Display the current IP address and netmask for all interfaces.

```
[student@servera ~]$ ip addr
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default
    qlen 1000
        link/loopback 00:00:00:00:00:00 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: enX: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group
    default qlen 1000
        link/ether 52:54:00:00:fa:0a brd ff:ff:ff:ff:ff:ff
        inet 172.25.250.10/24 brd 172.25.250.255 scope global noprefixroute ens3
            valid_lft forever preferred_lft forever
            inet6 fe80::3059:5462:198:58b2/64 scope link noprefixroute
                valid_lft forever preferred_lft forever
```

- 4. Display the statistics for the **enX** interface.

```
[student@servera ~]$ ip -s link show enX
2: enX: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP mode
    DEFAULT group default qlen 1000
        link/ether 52:54:00:00:fa:0a brd ff:ff:ff:ff:ff:ff
        RX: bytes packets errors dropped overrun mcast
            89014225 168251 0 154418 0 0
        TX: bytes packets errors dropped carrier collsns
            608808 6090 0 0 0 0
```

- 5. Display the routing information.

```
[student@servera ~]$ ip route
default via 172.25.250.254 dev enX proto static metric 100
172.25.250.0/24 dev enX proto kernel scope link src 172.25.250.10 metric 100
```

- 6. Verify that the router is accessible.

```
[student@servera ~]$ ping -c3 172.25.250.254
PING 172.25.250.254 (172.25.250.254) 56(84) bytes of data.
64 bytes from 172.25.250.254: icmp_seq=1 ttl=64 time=0.196 ms
64 bytes from 172.25.250.254: icmp_seq=2 ttl=64 time=0.436 ms
64 bytes from 172.25.250.254: icmp_seq=3 ttl=64 time=0.361 ms

--- 172.25.250.254 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 49ms
rtt min/avg/max/mdev = 0.196/0.331/0.436/0.100 ms
```

- 7. Show all the hops between the local system and **classroom.example.com**.

```
[student@servera ~]$ tracepath classroom.example.com
1?: [LOCALHOST]                                pmtu 1500
1:  workstation.lab.example.com                0.270ms
1:  workstation.lab.example.com                0.167ms
2:  classroom.example.com                     0.473ms reached
Resume: pmtu 1500 hops 2 back 2
```

- 8. Display the listening TCP sockets on the local system.

```
[student@servera ~]$ ss -lt
State      Recv-Q Send-Q      Local Address:Port      Peer Address:Port
LISTEN      0      128          0.0.0.0:sunrpc      0.0.0.0:*
LISTEN      0      128          0.0.0.0:ssh       0.0.0.0:*
LISTEN      0      128          [::]:sunrpc        [::]:*
LISTEN      0      128          [::]:ssh           [::]:*
```

- 9. Exit from **servera**.

```
[student@servera ~]$ exit
logout
Connection to servera closed.
[student@workstation ~]$
```

Finish

On **workstation**, run the **lab net-validate finish** script to finish this exercise.

```
[student@workstation ~]$ lab net-validate finish
```

This concludes the guided exercise.

Configuring Networking from the Command Line

Objectives

After completing this section, you should be able to manage network settings and devices using the **nmcli** command.

Describing NetworkManager Concepts

NetworkManager is a daemon that monitors and manages network settings. In addition to the daemon, there is a GNOME Notification Area applet providing 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 collection of settings that can be configured for a device.
- Only one connection can be *active* for any one device at a time. Multiple connections may exist for use by different devices or to allow a configuration to be altered for the same device. If you need to temporarily change networking settings, instead of changing the configuration of a connection, you can change which connection is active for a device. For example, a device for a wireless network interface on a laptop might use different connections for the wireless network at a work site and for the wireless network at home.
- Each connection has a *name* or ID that identifies it.
- The **nmcli** utility is used to create and edit connection files from the command line.

Viewing Networking Information

The **nmcli dev status** command displays the status of all network devices:

```
[user@host ~]$ nmcli dev status
DEVICE  TYPE      STATE       CONNECTION
eno1    ethernet  connected   eno1
ens3    ethernet  connected   static-ens3
eno2    ethernet  disconnected --
lo     loopback  unmanaged   --
```

The **nmcli con show** command displays a list of all connections. To list only the active connections, add the **--active** option.

```
[user@host ~]$ nmcli con show
NAME          UUID                                  TYPE      DEVICE
eno2          ff9f7d69-db83-4fed-9f32-939f8b5f81cd 802-3-ethernet --
static-ens3   72ca57a2-f780-40da-b146-99f71c431e2b 802-3-ethernet ens3
eno1          87b53c56-1f5d-4a29-a869-8a7bdaf56dfa 802-3-ethernet eno1
[user@host ~]$ nmcli con show --active
NAME          UUID                                  TYPE      DEVICE
```

```
static-ens3 72ca57a2-f780-40da-b146-99f71c431e2b 802-3-ethernet ens3
eno1        87b53c56-1f5d-4a29-a869-8a7bdaf56dfa 802-3-ethernet eno1
```

Adding a network connection

The **nmcli con add** command is used to add new network connections. The following example **nmcli con add** commands assume that the name of the network connection being added is not already in use.

The following command adds a new connection named **eno2** for the interface **eno2**, which gets IPv4 networking information using DHCP and autoconnects on startup. It also gets IPv6 networking settings by listening for router advertisements on the local link. The name of the configuration file is based on the value of the **con-name** option, **eno2**, and is saved to the **/etc/sysconfig/network-scripts/ifcfg-eno2** file.

```
[root@host ~]# nmcli con add con-name eno2 type ethernet ifname eno2
```

The next example creates an **eno2** connection for the **eno2** device with a static IPv4 address, using the IPv4 address and network prefix **192.168.0.5/24** and default gateway **192.168.0.254**, but still autoconnects at startup and saves its configuration into the same file. Due to screen size limitations, terminate the first line with a shell \ escape and complete the command on the next line.

```
[root@host ~]# nmcli con add con-name eno2 type ethernet ifname eno2 \
    ipv4.address 192.168.0.5/24 ipv4.gateway 192.168.0.254
```

This final example creates an **eno2** connection for the **eno2** device with static IPv6 and IPv4 addresses, using the IPv6 address and network prefix **2001:db8:0:1::c000:207/64** and default IPv6 gateway **2001:db8:0:1::1**, and the IPv4 address and network prefix **192.0.2.7/24** and default IPv4 gateway **192.0.2.1**, but still autoconnects at startup and saves its configuration into **/etc/sysconfig/network-scripts/ifcfg-eno2**. Due to screen size limitations, terminate the first line with a shell \ escape and complete the command on the next line.

```
[root@host ~]# nmcli con add con-name eno2 type ethernet ifname eno2 \
    ipv6.address 2001:db8:0:1::c000:207/64 ipv6.gateway 2001:db8:0:1::1 \
    ipv4.address 192.0.2.7/24 ipv4.gateway 192.0.2.1
```

Controlling network connections

The **nmcli con up name** command activates the connection *name* on the network interface it is bound to. Note that the command takes the name of a *connection*, not the name of the network interface. Remember that the **nmcli con show** command displays the names of *all* available connections.

```
[root@host ~]# nmcli con up static-ens3
```

The **nmcli dev disconnect device** command disconnects the network interface *device* and brings it down. This command can be abbreviated **nmcli dev dis device**:

```
[root@host ~]# nmcli dev dis ens3
```

**Important**

Use **nmcli dev dis device** to deactivate a network interface.

The **nmcli con down name** command is normally not the best way to deactivate a network interface because it brings down the connection. However, by default, most wired system connections are configured with **autoconnect** enabled. This activates the connection as soon as its network interface is available. Since the connection's network interface is still available, **nmcli con down name** brings the interface down, but then NetworkManager immediately brings it up again unless the connection is entirely disconnected from the interface.

Modifying Network Connection Settings

NetworkManager connections have two kinds of settings. There are *static* connection properties, configured by the administrator and stored in the configuration files in **/etc/sysconfig/network-scripts/ifcfg-***. There may also be *active* connection data, which the connection gets from a DHCP server and which are not stored persistently.

To list the current settings for a connection, run the **nmcli con show name** command, where *name* is the name of the connection. Settings in lowercase are static properties that the administrator can change. Settings in all caps are active settings in temporary use for this instance of the connection.

```
[root@host ~]# nmcli con show static-ens3
connection.id:                      static-ens3
connection.uuid:                     87b53c56-1f5d-4a29-a869-8a7bdaf56dfa
connection.interface-name:           --
connection.type:                     802-3-ethernet
connection.autoconnect:              yes
connection.timestamp:                1401803453
connection.read-only:                no
connection.permissions:              --
connection.zone:                     --
connection.master:                  --
connection.slave-type:               --
connection.secondaries:              --
connection.gateway-ping-timeout:    0
802-3-ethernet.port:                --
802-3-ethernet.speed:               0
802-3-ethernet.duplex:              --
802-3-ethernet.auto-negotiate:     yes
802-3-ethernet.mac-address:         CA:9D:E9:2A:CE:F0
802-3-ethernet.cloned-mac-address: --
802-3-ethernet.mac-address-blacklist: --
802-3-ethernet.mtu:                 auto
802-3-ethernet.s390-subchannels:   --
802-3-ethernet.s390-nettype:        --
802-3-ethernet.s390-options:       --
ipv4.method:                        manual
ipv4.dns:                           192.168.0.254
ipv4.dns-search:                   example.com
ipv4.addresses:                    { ip = 192.168.0.2/24, gw =
                                         192.168.0.254 }
```

```
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: manual  
ipv6.dns: 2001:4860:4860::8888  
ipv6.dns-search: example.com  
ipv6.addresses: { ip = 2001:db8:0:1::7/64, gw =  
    2001:db8:0:1::1 }  
ipv6.routes:  
ipv6.ignore-auto-routes: no  
ipv6.ignore-auto-dns: no  
ipv6.never-default: no  
ipv6.may-fail: yes  
ipv6.ip6-privacy: -1 (unknown)  
ipv6.dhcp-hostname: --  
...output omitted...
```

The **nmcli con mod name** command is used to change the settings for a connection. These changes are also saved in the **/etc/sysconfig/network-scripts/ifcfg-name** file for the connection. Available settings are documented in the **nm-settings(5)** man page.

To set the IPv4 address to **192.0.2.2/24** and default gateway to **192.0.2.254** for the connection **static-ens3**:

```
[root@host ~]# nmcli con mod static-ens3 ipv4.address 192.0.2.2/24 \  
    ipv4.gateway 192.0.2.254
```

To set the IPv6 address to **2001:db8:0:1::a00:1/64** and default gateway to **2001:db8:0:1::1** for the connection **static-ens3**:

```
[root@host ~]# nmcli con mod static-ens3 ipv6.address 2001:db8:0:1::a00:1/64 \  
    ipv6.gateway 2001:db8:0:1::1
```



Important

If a connection that gets its IPv4 information from a DHCPv4 server is being changed to get it from static configuration files only, the setting **ipv4.method** should also be changed from **auto** to **manual**.

Likewise, if a connection that gets its IPv6 information by SLAAC or a DHCPv6 server is being changed to get it from static configuration files only, the setting **ipv6.method** should also be changed from **auto** or **dhcp** to **manual**.

Otherwise, the connection may hang or not complete successfully when it is activated, or it may get an IPv4 address from DHCP or an IPv6 address from DHCPv6 or SLAAC in addition to the static address.

A number of settings may have multiple values. A specific value can be added to the list or deleted from the list for a setting by adding a + or - symbol to the start of the setting name.

Deleting a network connection

The **nmcli con del name** command deletes the connection named *name* from the system, disconnecting it from the device and removing the file **/etc/sysconfig/network-scripts/ifcfg-name**.

```
[root@host ~]# nmcli con del static-ens3
```

Who Can Modify Network Settings?

The **root** user can make any necessary network configuration changes with **nmcli**.

However, regular users that are logged in on the local console can also make many network configuration changes to the system. They have to log in at the system's keyboard to either a text-based virtual console or the graphical desktop environment to get this control. The logic behind this is that if someone is physically present at the computer's console, it's likely being used as a workstation or laptop and they may need to configure, activate, and deactivate wireless or wired network interfaces at will. By contrast, if the system is a server in the datacenter, generally the only users logging in locally to the machine itself should be administrators.

Regular users that log in using **ssh** do not have access to change network permissions without becoming **root**.

You can use the **nmcli gen permissions** command to see what your current permissions are.

Summary of Commands

The following table is a list of key **nmcli** commands discussed in this section.

Command	Purpose
nmcli dev status	Show the NetworkManager status of all network interfaces.
nmcli con show	List all connections.
nmcli con show name	List the current settings for the connection <i>name</i> .
nmcli con add con-name name	Add a new connection named <i>name</i> .
nmcli con mod name	Modify the connection <i>name</i> .
nmcli con reload	Reload the configuration files (useful after they have been edited by hand).
nmcli con up name	Activate the connection <i>name</i> .
nmcli dev dis dev	Deactivate and disconnect the current connection on the network interface <i>dev</i> .
nmcli con del name	Delete the connection <i>name</i> and its configuration file.



References

NetworkManager(8), **nmcli(1)**, **nmcli-examples(5)**, **nm-settings(5)**,
hostnamectl(1), **resolv.conf(5)**, **hostname(5)**, **ip(8)**, and **ip-address(8)**
man pages

► Guided Exercise

Configuring Networking from the Command Line

In this exercise, you will configure network settings using **nmcli**.

Outcomes

You should be able to convert a system from DHCP to static configuration.

Before You Begin

Log in as the **student** user on **workstation** using **student** as the password.

From **workstation**, run the **lab net-configure start** command. The command runs a start script that determine if the host, **servera**, is reachable on the network.

```
[student@workstation ~]$ lab net-configure start
```



Note

If prompted by the **sudo** command for **student**'s password, enter **student** as the password.

- 1. Use the **ssh** command to log in to **servera** as the **student** user. The systems are configured to use SSH keys for authentication, so a password is not required to log in to **servera**.

```
[student@workstation ~]$ ssh student@servera  
...output omitted...  
[student@servera ~]$
```

- 2. Locate network interface names.



Important

Network interface names are determined by their bus type and the detection order of devices during boot. Your network interface names will vary according to the course platform and hardware in use.

On your system now, locate the interface name (such as **ens06** or **en1p2**) associated with the Ethernet address **52:54:00:00:fa:0a**. Use this interface name to replace the **enX** placeholder used throughout this exercise.

Locate the network interface name associated with the Ethernet address **52:54:00:00:fa:0a**. Record or remember this name and use it to replace the **enX** placeholder in subsequent commands.

```
[student@servera ~]$ ip link
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default
    qlen 1000
        link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
2: enX: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default
    qlen 1000
        link/ether 52:54:00:00:fa:0a brd ff:ff:ff:ff:ff:ff
```

► 3. View network settings using **nmcli**.

3.1. Show all connections.

```
[student@servera ~]$ nmcli con show
NAME           UUID                                  TYPE      DEVICE
Wired connection 1  03da038a-3257-4722-a478-53055cc90128  ethernet  enX
```

3.2. Display only the active connection.

Your network interface name should appear under **DEVICE**, and the name of the connection active for that device is listed on the same line under **NAME**. This exercise assumes that the active connection is **Wired connection 1**.

If the name of the active connection is different, use that instead of **Wired connection 1** for the rest of this exercise.

```
[student@servera ~]$ nmcli con show --active
NAME           UUID                                  TYPE      DEVICE
Wired connection 1  03da038a-3257-4722-a478-53055cc90128  ethernet  enX
```

3.3. Display all configuration settings for the active connection.

```
[student@servera ~]$ nmcli con show "Wired connection 1"
connection.id:            Wired connection 1
connection.uuid:          03da038a-3257-4722-a478-53055cc90128
connection.stable-id:     --
connection.type:          802-3-ethernet
connection.interface-name: --
connection.autoconnect:   yes
...output omitted...
ipv4.method:              manual
ipv4.dns:                 172.25.250.254
ipv4.dns-search:          lab.example.com,example.com
ipv4.dns-options:         ""
ipv4.dns-priority:        0
ipv4.addresses:           172.25.250.10/24
ipv4.gateway:             172.25.250.254
...output omitted...
GENERAL.NAME:             Wired connection 1
GENERAL.UUID:              03da038a-3257-4722-a478-53055cc90128
```

```

GENERAL.DEVICES:          enX
GENERAL.STATE:            activated
GENERAL.DEFAULT:          yes
GENERAL.DEFAULT6:         no
GENERAL.SPEC-OBJECT:      --
GENERAL.VPN:              no
GENERAL.DBUS-PATH:        /org/freedesktop/NetworkManager/ActiveConnection/1
GENERAL.CON-PATH:          /org/freedesktop/NetworkManager/Settings/1
GENERAL.ZONE:              --
GENERAL.MASTER-PATH:       --
IP4.ADDRESS[1]:           172.25.250.10/24
IP4.GATEWAY:              172.25.250.254
IP4.ROUTE[1]:              dst = 172.25.250.0/24, nh = 0.0.0.0, mt = 100
IP4.ROUTE[2]:              dst = 0.0.0.0/0, nh = 172.25.250.254, mt = 100
IP4.DNS[1]:                172.25.250.254
IP6.ADDRESS[1]:            fe80::3059:5462:198:58b2/64
IP6.GATEWAY:              --
IP6.ROUTE[1]:              dst = fe80::/64, nh = ::, mt = 100
IP6.ROUTE[2]:              dst = ff00::/8, nh = ::, mt = 256, table=255

```

Press **q** to exit the command.

3.4. Show device status.

```
[student@servera ~]$ nmcli dev status
DEVICE  TYPE      STATE      CONNECTION
enX     ethernet  connected  Wired connection 1
lo      loopback  unmanaged  --
```

3.5. Display the settings for the **enX** device.

```
[student@servera ~]$ nmcli dev show enX
GENERAL.DEVICE:          enX
GENERAL.TYPE:             ethernet
GENERAL.HWADDR:           52:54:00:00:FA:0A
GENERAL.MTU:               1500
GENERAL.STATE:             100 (connected)
GENERAL.CONNECTION:        Wired connection 1
GENERAL.CON-PATH:          /org/freedesktop/NetworkManager/ActiveConnection/1
WIRED-PROPERTIES.CARRIER: on
IP4.ADDRESS[1]:            172.25.250.10/24
IP4.GATEWAY:              172.25.250.254
IP4.ROUTE[1]:              dst = 172.25.250.0/24, nh = 0.0.0.0, mt = 100
IP4.ROUTE[2]:              dst = 0.0.0.0/0, nh = 172.25.250.254, mt = 100
IP4.DNS[1]:                172.25.250.254
IP6.ADDRESS[1]:            fe80::3059:5462:198:58b2/64
IP6.GATEWAY:              --
IP6.ROUTE[1]:              dst = fe80::/64, nh = ::, mt = 100
IP6.ROUTE[2]:              dst = ff00::/8, nh = ::, mt = 256, table=255
```

- ▶ 4. Create a static connection with the same IPv4 address, network prefix, and default gateway.
Name the new connection *static-addr*.

**Warning**

Since access to your machine is provided over the primary network connection, setting incorrect values during network configuration may make your machine unreachable. If this happens, use the **Reset** button located above what used to be your machine's graphical display and try again.

```
[student@servera ~]$ sudo nmcli con add con-name "static-addr" ifname enX \
type ethernet ipv4.method manual \
ipv4.address 172.25.250.10/24 ipv4.gateway 172.25.250.254
Connection 'static-addr' (15aa3901-555d-40cb-94c6-cea6f9151634) successfully
added.
```

- ▶ 5. Modify the new connection to add the DNS setting.

```
[student@servera ~]$ sudo nmcli con mod "static-addr" ipv4.dns 172.25.250.254
```

- ▶ 6. Display and activate the new connection.

- 6.1. View all connections.

```
[student@servera ~]$ nmcli con show
NAME           UUID                                  TYPE      DEVICE
Wired connection 1  03da038a-3257-4722-a478-53055cc90128  ethernet  enX
static-addr      15aa3901-555d-40cb-94c6-cea6f9151634  ethernet  --
```

- 6.2. View the active connection.

```
[student@servera ~]$ nmcli con show --active
NAME           UUID                                  TYPE      DEVICE
Wired connection 1  03da038a-3257-4722-a478-53055cc90128  ethernet  enX
```

- 6.3. Activate the new **static-addr** connection.

```
[student@servera ~]$ sudo nmcli con up "static-addr"
Connection successfully activated (D-Bus active path: /org/freedesktop/
NetworkManager/ActiveConnection/2)
```

- 6.4. Verify the new active connection.

```
[student@servera ~]$ nmcli con show --active
NAME           UUID                                  TYPE      DEVICE
static-addr    15aa3901-555d-40cb-94c6-cea6f9151634  ethernet  enX
```

- ▶ 7. Configure the original connection so that it does not start at boot, and verify that the static connection is used when the system reboots.

- 7.1. Disable the original connection from autostarting at boot.

```
[student@servera ~]$ sudo nmcli con mod "Wired connection 1" \
connection.autoconnect no
```

7.2. Reboot the system.

```
[student@servera ~]$ sudo systemctl reboot
Connection to servera closed by remote host.
Connection to servera closed.
[student@workstation ~]$
```

7.3. View the active connection.

```
[student@workstation ~]$ ssh student@servera
...output omitted...
[student@servera ~]$ nmcli con show --active
NAME           UUID                                  TYPE      DEVICE
static-addr   15aa3901-555d-40cb-94c6-cea6f9151634  ethernet  enx
```

► 8. Test connectivity using the new network addresses.

8.1. Verify the IP address.

```
[student@servera ~]$ ip addr show enX
2: enX: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group
default qlen 1000
    link/ether 52:54:00:00:fa:0a brd ff:ff:ff:ff:ff:ff
    inet 172.25.250.10/24 brd 172.25.250.255 scope global noprefixroute enX
        valid_lft forever preferred_lft forever
    inet6 fe80::6556:cdd9:ce15:1484/64 scope link noprefixroute
        valid_lft forever preferred_lft forever
```

8.2. Verify the default gateway.

```
[student@servera ~]$ ip route
default via 172.25.250.254 dev enX proto static metric 100
172.25.250.0/24 dev enX proto kernel scope link src 172.25.250.10 metric 100
```

8.3. Ping the DNS address.

```
[student@servera ~]$ ping -c3 172.25.250.254
PING 172.25.250.254 (172.25.250.254) 56(84) bytes of data.
64 bytes from 172.25.250.254: icmp_seq=1 ttl=64 time=0.225 ms
64 bytes from 172.25.250.254: icmp_seq=2 ttl=64 time=0.314 ms
64 bytes from 172.25.250.254: icmp_seq=3 ttl=64 time=0.472 ms

--- 172.25.250.254 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 46ms
rtt min/avg/max/mdev = 0.225/0.337/0.472/0.102 ms
```

8.4. Exit from **servera**.

```
[student@servera ~]$ exit  
logout  
Connection to servera closed.  
[student@workstation ~]$
```

Finish

On **workstation**, run the **lab net-configure finish** script to finish this exercise.

```
[student@workstation ~]$ lab net-configure finish
```

This concludes the guided exercise.

Editing Network Configuration Files

Objectives

After completing this section, you should be able to modify network configuration by editing configuration files.

Describing Connection Configuration Files

By default, changes made with `nmcli con mod name` are automatically saved to `/etc/sysconfig/network-scripts/ifcfg-name`. That file can also be manually edited with a text editor. After doing so, run `nmcli con reload` so that NetworkManager reads the configuration changes.

For backward-compatibility reasons, the directives saved in that file have different names and syntax than the `nm-settings(5)` names. The following table maps some of the key setting names to `ifcfg-*` directives.

Comparison of nm-settings and ifcfg-* Directives

<code>nmcli con mod</code>	<code>ifcfg-* file</code>	<code>Effect</code>
<code>ipv4.method manual</code>	<code>BOOTPROTO=none</code>	IPv4 addresses configured statically.
<code>ipv4.method auto</code>	<code>BOOTPROTO=dhcp</code>	Looks for configuration settings from a DHCPv4 server. If static addresses are also set, will not bring those up until we have information from DHCPv4.
<code>ipv4.addresses 192.0.2.1/24</code>	<code>IPADDR=192.0.2.1</code> <code>PREFIX=24</code>	Sets static IPv4 address and network prefix. If more than one address is set for the connection, then the second one is defined by the <code>IPADDR1</code> and <code>PREFIX1</code> directives, the third one by the <code>IPADDR2</code> and <code>PREFIX2</code> directives, and so on.
<code>ipv4.gateway 192.0.2.254</code>	<code>GATEWAY=192.0.2.254</code>	Sets the default gateway.
<code>ipv4.dns 8.8.8.8</code>	<code>DNS1=8.8.8.8</code>	Modify <code>/etc/resolv.conf</code> to use this <code>nameserver</code> .

nmcli con mod	ifcfg-* file	Effect
ipv4.dns-search example.com	DOMAIN=example.com	Modify /etc/resolv.conf to use this domain in the search directive.
ipv4.ignore-auto-dns true	PEERDNS=no	Ignore DNS server information from the DHCP server.
ipv6.method manual	IPV6_AUTOCONF=no	IPv6 addresses configured statically.
ipv6.method auto	IPV6_AUTOCONF=yes	Configures network settings using SLAAC from router advertisements.
ipv6.method dhcp	IPV6_AUTOCONF=no DHCPV6C=yes	Configures network settings by using DHCPv6, but not SLAAC.
ipv6.addresses 2001:db8::a/64	IPV6ADDR=2001:db8::a/64	Sets static IPv6 address and network prefix. If more than one address is set for the connection, IPV6ADDR_SECONDARIES takes a double-quoted list of space-delimited address/prefix definitions.
ipv6.gateway 2001:db8::1	IPV6_DEFAULTGW=2001:...	Sets the default gateway.
ipv6.dns fde2:6494:1e09:2::d	DNS1=fde2:6494:... DNS2=fde2:6494:1e09:2::d	Modify /etc/resolv.conf to use this nameserver. Exactly the same as IPv4.
ipv6.dns-search example.com	IPV6_DOMAIN=example.com	Modify /etc/resolv.conf to use this domain in the search directive.
ipv6.ignore-auto-dns true	IPV6_PEERDNS=no	Ignore DNS server information from the DHCP server.
connection.autoconnect yes	ONBOOT=yes	Automatically activate this connection at boot.
connection.id ens3	NAME=ens3	The name of this connection.
connection.interface-name ens3	DEVICE=ens3	The connection is bound to the network interface with this name.

<code>nmcli con mod</code>	<code>ifcfg-* file</code>	<code>Effect</code>
<code>802-3-ethernet.mac-address ...</code>	<code>HWADDR=...</code>	The connection is bound to the network interface with this MAC address.

Modifying network configuration

It is also possible to configure the network by directly editing the connection configuration files. Connection 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 IPv4 configuration.

IPv4 Configuration Options for `ifcfg` File

<code>Static</code>	<code>Dynamic</code>	<code>Either</code>
<code>BOOTPROTO=none</code>	<code>BOOTPROTO=dhcp</code>	<code>DEVICE=ens3</code>
<code>IPADDR0=172.25.250.10</code>		<code>NAME="static-ens3"</code>
<code>PREFIX0=24</code>		<code>ONBOOT=yes</code>
<code>GATEWAY0=172.25.250.254</code>		<code>UUID=f3e8(...)ad3e</code>
<code>DEFROUTE=yes</code>		<code>USERCTL=yes</code>
<code>DNS1=172.25.254.254</code>		

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 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@host ~]# nmcli con reload
[root@host ~]# nmcli con down "static-ens3"
[root@host ~]# nmcli con up "static-ens3"
```



References

`nmcli(1)` man page

For more information, refer to the *Configuring and Managing Networking Guide* at
https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/8/html-single/configuring_and_managing_networking/

► Guided Exercise

Editing Network Configuration Files

In this exercise, you will manually modify network configuration files and ensure that the new settings take effect.

Outcomes

You should be able to add an additional network address to each system.

Before You Begin

Log in as the **student** user on **workstation** using **student** as the password.

From **workstation**, run the **lab net-edit start** command. The command runs a start script that determine if the hosts, **servera** and **serverb**, are reachable on the network.

```
[student@workstation ~]$ lab net-edit start
```

- ▶ 1. Use the **ssh** command to log in to **servera** as the **student** user. The systems are configured to use SSH keys for authentication, so a password is not required to log in to **servera**.

```
[student@workstation ~]$ ssh student@servera  
...output omitted...  
[student@servera ~]$
```

- ▶ 2. Locate network interface names.



Important

Network interface names are determined by their bus type and the detection order of devices during boot. Your network interface names will vary according to the course platform and hardware in use.

On your system now, locate the interface name (such as **ens06** or **en1p2**) associated with the Ethernet address **52:54:00:00:fa:0a**. Use this interface name to replace the **enX** placeholder used throughout this exercise.

Locate the network interface name associated with the Ethernet address **52:54:00:00:fa:0a**. Record or remember this name and use it to replace the **enX** placeholder in subsequent commands. The active connection is also named **Wired connection 1** (and therefore is managed by the file **/etc/sysconfig/network-scripts/ifcfg-Wired_connection_1**).

If you have done previous exercises in this chapter and this was true for your system, it should be true for this exercise as well.

```
[student@servera ~]$ ip link
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default
    qlen 1000
        link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
2: enX: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group
    default qlen 1000
        link/ether 52:54:00:00:fa:0a brd ff:ff:ff:ff:ff:ff
[student@servera ~]$ nmcli con show --active
NAME           UUID                                  TYPE      DEVICE
Wired connection 1  03da038a-3257-4722-a478-53055cc90128  ethernet  enX
[student@servera ~]$ ls \
/etc/sysconfig/network-scripts/ifcfg-Wired_connection_1
/etc/sysconfig/network-scripts/ifcfg-Wired_connection_1
```

- 3. On **servera**, switch to the **root** user, and then edit the **/etc/sysconfig/network-scripts/ifcfg-Wired_connection_1** file to add an additional address of **10.0.1.1/24**.

- 3.1. Use the **sudo -i** command to switch to the **root** user. The password for the **student** user is **student**.

```
[student@servera ~]$ sudo -i
[sudo] password for student: student
[root@servera ~]#
```

- 3.2. Append an entry to the file to specify the IPv4 address.

```
[root@servera ~]# echo \
"IPADDR1=10.0.1.1" >> /etc/sysconfig/network-scripts/ifcfg-Wired_connection_1
```

- 3.3. Append an entry to the file to specify the network prefix.

```
[root@servera ~]# echo \
"PREFIX1=24" >> /etc/sysconfig/network-scripts/ifcfg-Wired_connection_1
```

- 4. Activate the new address.

- 4.1. Reload the configuration changes.

```
[root@servera ~]# nmcli con reload
```

- 4.2. Restart the connection with the new settings.

```
[root@servera ~]# nmcli con up "Wired connection 1"
Connection successfully activated (D-Bus active path: /org/freedesktop/
NetworkManager/ActiveConnection/3)
```

- 5. Verify the new IP address.

```
[root@servera ~]# ip addr show enX
2: enX: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group
    default qlen 1000
        link/ether 52:54:00:00:fa:0a brd ff:ff:ff:ff:ff:ff
        inet 172.25.250.10/24 brd 172.25.250.255 scope global noprefixroute enX
            valid_lft forever preferred_lft forever
        inet 10.0.1.1/24 brd 10.0.1.255 scope global noprefixroute enX
            valid_lft forever preferred_lft forever
        inet6 fe80::4bf3:e1d9:3076:f8d7/64 scope link noprefixroute
            valid_lft forever preferred_lft forever
```

- 6. Exit from **servera** to return to **workstation** as the **student** user.

```
[root@servera ~]# exit
logout
[student@servera ~]$ exit
logout
Connection to servera closed.
[student@workstation ~]$
```

- 7. On **serverb**, edit the **/etc/sysconfig/network-scripts/ifcfg-Wired_connection_1** file to add an additional address of **10.0.1.2/24**, then load the new configuration.

- 7.1. From **workstation**, use the **ssh** command to log in to **serverb** as the **student** user.

```
[student@workstation ~]$ ssh student@serverb
...output omitted...
[student@serverb ~]$
```

- 7.2. Use the **sudo -i** command to switch to the **root** user. The password for the **student** user is **student**.

```
[student@serverb ~]$ sudo -i
[sudo] password for student: student
[root@serverb ~]#
```

- 7.3. Modify the **ifcfg-Wired_connection_1** file to add the second IPv4 address and network prefix.

```
[root@serverb ~]# echo \
"IPADDR1=10.0.1.2" >> /etc/sysconfig/network-scripts/ifcfg-Wired_connection_1
[root@serverb ~]# echo \
"PREFIX1=24" >> /etc/sysconfig/network-scripts/ifcfg-Wired_connection_1
```

- 7.4. Reload the configuration changes.

```
[root@serverb ~]# nmcli con reload
```

7.5. Bring up the connection with the new settings.

```
[root@serverb ~]# nmcli con up "Wired connection 1"
Connection successfully activated (D-Bus active path: /org/freedesktop/
NetworkManager/ActiveConnection/4)
```

7.6. Verify the new IP address.

```
[root@serverb ~]# ip addr show enX
2: enX: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group
    default qlen 1000
        link/ether 52:54:00:00:fa:0b brd ff:ff:ff:ff:ff:ff
        inet 172.25.250.11/24 brd 172.25.250.255 scope global noprefixroute enX
            valid_lft forever preferred_lft forever
        inet 10.0.1.2/24 brd 10.0.1.255 scope global noprefixroute enX
            valid_lft forever preferred_lft forever
        inet6 fe80::74c:3476:4113:463f/64 scope link noprefixroute
            valid_lft forever preferred_lft forever
```

► 8. Test connectivity using the new network addresses.

8.1. From **serverb**, ping the new address of **servera**.

```
[root@serverb ~]# ping -c3 10.0.1.1
PING 10.0.1.1 (10.0.1.1) 56(84) bytes of data.
64 bytes from 10.0.1.1: icmp_seq=1 ttl=64 time=0.342 ms
64 bytes from 10.0.1.1: icmp_seq=2 ttl=64 time=0.188 ms
64 bytes from 10.0.1.1: icmp_seq=3 ttl=64 time=0.317 ms

--- 10.0.1.1 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 35ms
rtt min/avg/max/mdev = 0.188/0.282/0.342/0.068 ms
```

8.2. Exit from **serverb** to return to **workstation**.

```
[root@serverb ~]# exit
logout
[student@serverb ~]$ exit
logout
Connection to serverb closed.
[student@workstation ~]$
```

8.3. From **workstation**, use the **ssh** command to access **servera** as the **student** user to ping the new address of **serverb**.

```
[student@workstation ~]$ ssh student@servera ping -c3 10.0.1.2
PING 10.0.1.2 (10.0.1.2) 56(84) bytes of data.
64 bytes from 10.0.1.2: icmp_seq=1 ttl=64 time=0.269 ms
64 bytes from 10.0.1.2: icmp_seq=2 ttl=64 time=0.338 ms
64 bytes from 10.0.1.2: icmp_seq=3 ttl=64 time=0.361 ms
```

```
--- 10.0.1.2 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 48ms
rtt min/avg/max/mdev = 0.269/0.322/0.361/0.044 ms
```

Finish

On **workstation**, run the **lab net-edit finish** script to finish this exercise.

```
[student@workstation ~]$ lab net-edit finish
```

This concludes the guided exercise.

Configuring Host Names and Name Resolution

Objectives

After completing this section, you should be able to configure a server's static host name and its name resolution and test the results.

Changing the system host name

The **hostname** command displays or temporarily modifies the system's fully qualified host name.

```
[root@host ~]# hostname
host.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@host ~]# hostnamectl set-hostname host.example.com
[root@host ~]# hostnamectl status
  Static hostname: host.example.com
    Icon name: computer-vm
      Chassis: vm
    Machine ID: f874df04639f474cb0a9881041f4f7d4
      Boot ID: 6a0abe03ef0b4a97bcb8afb7b281e4d3
  Virtualization: kvm
Operating System: Red Hat Enterprise Linux 8.2 (Ootpa)
      CPE OS Name: cpe:/o:redhat:enterprise_linux:8.2:GA
        Kernel: Linux 4.18.0-193.el8.x86_64
      Architecture: x86-64
[root@host ~]# cat /etc/hostname
host.example.com
```



Important

In Red Hat Enterprise Linux 7 and later, the static host name is stored in **/etc/hostname**. Red Hat Enterprise Linux 6 and earlier stores 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. It determines where to look based on the configuration of the **/etc/nsswitch.conf** file. By default, the contents of the **/etc/hosts** file are checked first.

```
[root@host ~]# 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 using the **/etc/hosts** file.

If an entry is not found in the **/etc/hosts** file, by default the stub resolver tries to look up the hostname by using a DNS nameserver. The **/etc/resolv.conf** file controls how this query is performed:

- **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.
- **nameserver**: the IP address of a nameserver to query. Up to three nameserver directives may be given to provide backups if one is down.

```
[root@host ~]# cat /etc/resolv.conf
# Generated by NetworkManager
domain example.com
search example.com
nameserver 172.25.254.254
```

NetworkManager updates the **/etc/resolv.conf** file using DNS settings in the connection configuration files. Use the **nmcli** command to modify the connections.

```
[root@host ~]# nmcli con mod ID ipv4.dns IP
[root@host ~]# nmcli con down ID
[root@host ~]# nmcli con up ID
[root@host ~]# cat /etc/sysconfig/network-scripts/ifcfg-ID
...output omitted...
DNS1=8.8.8.8
...output omitted...
```

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 + or - symbol in front of the **ipv4.dns** argument adds or removes an individual entry.

```
[root@host ~]# nmcli con mod ID +ipv4.dns IP
```

To add the DNS server with IPv6 IP address **2001:4860:4860::8888** to the list of nameservers to use with the connection **static-ens3**:

```
[root@host ~]# nmcli con mod static-ens3 +ipv6.dns 2001:4860:4860::8888
```

**Note**

Static IPv4 and IPv6 DNS settings all end up as **nameserver** directives in **/etc/resolv.conf**. You should ensure that there is, at minimum, an IPv4-reachable name server listed (assuming a dual-stack system). It is better to have at least one name server using IPv4 and a second using IPv6 in case you have network issues with either your IPv4 or IPv6 networking.

Testing DNS Name Resolution

The **host HOSTNAME** command can be used to test DNS server connectivity.

```
[root@host ~]# host classroom.example.com
classroom.example.com has address 172.25.254.254
[root@host ~]# host 172.25.254.254
254.254.25.172.in-addr.arpa domain name pointer classroom.example.com.
```

**Important**

DHCP automatically rewrites the **/etc/resolv.conf** file as interfaces are started unless you specify **PEERDNS=no** in the relevant interface configuration files. Set this using the **nmcli** command.

```
[root@host ~]# nmcli con mod "static-ens3" ipv4.ignore-auto-dns yes
```

**References**

nmcli(1), **hostnamectl(1)**, **hosts(5)**, **getent(1)**, **host(1)**, and **resolv.conf(5)** man pages

For more information, refer to the *Configuring and Managing Networking Guide* at https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/8/html-single/configuring_and_managing_networking/

► Guided Exercise

Configuring Host Names and Name Resolution

In this exercise, you will manually configure the system’s static host name, `/etc/hosts` file, and DNS name resolver.

Outcomes

You should be able to set a customized host name and configure name resolution settings.

Before You Begin

Log in as the **student** user on **workstation** using **student** as the password.

From **workstation**, run the **lab net-hostnames start** command. The command runs a start script that determine if the host, **servera**, is reachable on the network.

```
[student@workstation ~]$ lab net-hostnames start
```

- 1. Use the **ssh** command to log in to **servera** as the **student** user. The systems are configured to use SSH keys for authentication, so a password is not required to log in to **servera**.

```
[student@workstation ~]$ ssh student@servera  
...output omitted...  
[student@servera ~]$
```

- 2. View the current host name settings.

- 2.1. Display the current host name.

```
[student@servera ~]$ hostname  
servera.lab.example.com
```

- 2.2. Display the host name status.

```
[student@servera ~]$ hostnamectl status  
  Static hostname: n/a  
Transient hostname: servera.lab.example.com  
    Icon name: computer-vm  
      Chassis: vm  
Machine ID: f874df04639f474cb0a9881041f4f7d4  
   Boot ID: 22ae5279f57049678eda547bdb39a19d  
Virtualization: kvm  
Operating System: Red Hat Enterprise Linux 8.2 (Ootpa)
```

```
CPE OS Name: cpe:/o:redhat:enterprise_linux:8.2:GA  
Kernel: Linux 4.18.0-193.el8.x86_64  
Architecture: x86-64
```

Note the transient hostname obtained from DHCP or mDNS.

► 3. Set a static host name to match the current transient host name.

3.1. Change the host name and host name configuration file.

```
[student@servera ~]$ sudo hostnamectl set-hostname \  
servera.lab.example.com  
[sudo] password for student: student  
[student@servera ~]$
```

3.2. View the content of the **/etc/hostname** file which provides the host name at network start.

```
servera.lab.example.com
```

3.3. Display the host name status.

```
[student@servera ~]$ hostnamectl status  
Static hostname: servera.lab.example.com  
Icon name: computer-vm  
Chassis: vm  
Machine ID: f874df04639f474cb0a9881041f4f7d4  
Boot ID: 22ae5279f57049678eda547bdb39a19d  
Virtualization: kvm  
Operating System: Red Hat Enterprise Linux 8.2 (Ootpa)  
CPE OS Name: cpe:/o:redhat:enterprise_linux:8.2:GA  
Kernel: Linux 4.18.0-193.el8.x86_64  
Architecture: x86-64
```

Note that the transient hostname is not shown now that a static hostname has been configured.

► 4. Temporarily change the host name.

4.1. Change the host name.

```
[student@servera ~]$ sudo hostname testname
```

4.2. Display the current host name.

```
[student@servera ~]$ hostname  
testname
```

4.3. View the content of the **/etc/hostname** file which provides the host name at network start.

```
servera.lab.example.com
```

4.4. Reboot the system.

```
[student@servera ~]$ sudo systemctl reboot
Connection to servera closed by remote host.
Connection to servera closed.
[student@workstation ~]$
```

4.5. From **workstation** log in to **servera** as **student** user.

```
[student@workstation ~]$ ssh student@servera
...output omitted...
[student@servera ~]$
```

4.6. Display the current host name.

```
[student@servera ~]$ hostname
servera.lab.example.com
```

▶ 5. Add a local nickname for the classroom server.

5.1. Look up the IP address of the classroom.example.com.

```
[student@servera ~]$ host classroom.example.com
classroom.example.com has address 172.25.254.254
```

5.2. Modify **/etc/hosts** so that the additional name of **class** can be used to access the IP address 172.25.254.254.

```
127.0.0.1    localhost localhost.localdomain localhost4 localhost4.localdomain4
::1          localhost localhost.localdomain localhost6 localhost6.localdomain6

172.25.254.254 classroom.example.com classroom class
172.25.254.254 content.example.com content
...content omitted...
```

5.3. Look up the IP address of **class**.

```
[student@servera ~]$ host class
Host class not found: 2(SERVFAIL)
[student@servera ~]$ getent hosts class
172.25.254.254    classroom.example.com class
```

5.4. Ping **class**.

```
[student@servera ~]$ 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
```

5.5. Exit from **servera**.

```
[student@servera ~]$ exit
logout
Connection to servera closed.
[student@workstation ~]$
```

Finish

On **workstation**, run the **lab net-hostnames finish** script to finish this exercise.

```
[student@workstation ~]$ lab net-hostnames finish
```

This concludes the guided exercise.

▶ Lab

Managing Networking

Performance Checklist

In this lab, you will configure networking settings on a Red Hat Enterprise Linux server.

Outcomes

You should be able to configure two static IPv4 addresses for the primary network interface.

Before You Begin

Log in as the **student** user on **workstation** using **student** as the password.

From **workstation** run the **lab net-review start** command. The command runs a start script that determine if the host, **serverb**, is reachable on the network.

```
[student@workstation ~]$ lab net-review start
```

1. Use the **ssh** command to log in to **serverb** as the **student** user. The systems are configured to use SSH keys for authentication, so a password is not required to log in to **serverb**.
2. Use the **sudo -i** command to switch to the **root** user. If prompted, use **student** as the password.
3. Create a new connection with a static network connection using the settings in the table.

Parameter	Setting
Connection name	lab
Interface name	enX (might vary, use the interface that has 52:54:00:00:fa:0b as its MAC address)
IP address	172.25.250.11/24
Gateway address	172.25.250.254
DNS address	172.25.250.254

4. Configure the new connection to be autostarted. Other connections should not start automatically.
5. Modify the new connection so that it also uses the address 10.0.1.1/24.
6. Configure the **hosts** file so that 10.0.1.1 can be referenced as **private**.
7. Reboot the system.
8. From **workstation** use the **ping** command to verify that **serverb** is initialized.

Evaluation

On workstation, run the **lab net-review grade** script to confirm success on this lab.

```
[student@workstation ~]$ lab net-review grade
```

Finish

On **workstation**, run the **lab net-review finish** script to finish this lab.

```
[student@workstation ~]$ lab net-review finish
```

This concludes the lab.

► Solution

Managing Networking

Performance Checklist

In this lab, you will configure networking settings on a Red Hat Enterprise Linux server.

Outcomes

You should be able to configure two static IPv4 addresses for the primary network interface.

Before You Begin

Log in as the **student** user on **workstation** using **student** as the password.

From **workstation** run the **lab net-review start** command. The command runs a start script that determine if the host, **serverb**, is reachable on the network.

```
[student@workstation ~]$ lab net-review start
```

1. Use the **ssh** command to log in to **serverb** as the **student** user. The systems are configured to use SSH keys for authentication, so a password is not required to log in to **serverb**.

```
[student@workstation ~]$ ssh student@serverb
...output omitted...
[student@serverb ~]$
```

2. Use the **sudo -i** command to switch to the **root** user. If prompted, use **student** as the password.

```
[student@serverb ~]$ sudo -i
[sudo] password for student: student
[root@serverb ~]#
```

3. Create a new connection with a static network connection using the settings in the table.

Parameter	Setting
Connection name	lab
Interface name	enX (might vary, use the interface that has 52:54:00:00:fa:0b as its MAC address)
IP address	172.25.250.11/24
Gateway address	172.25.250.254
DNS address	172.25.250.254

Determine the interface name and the current active connection's name. The solution assumes that the interface name is **enX** and the connection name is **Wired connection 1**.

```
[root@serverb ~]# ip link
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default
    qlen 1000
        link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
2: enX: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default
    qlen 1000
        link/ether 52:54:00:00:fa:0b brd ff:ff:ff:ff:ff:ff
[root@serverb ~]# nmcli con show --active
NAME           UUID                                  TYPE      DEVICE
Wired connection 1  03da038a-3257-4722-a478-53055cc90128  ethernet  enX
```

Create the new **lab** connection profile based on the information in the table described in the instructions. Associate the profile with your network interface name listed in the output of the previous **ip link** command.

```
[root@serverb ~]# nmcli con add con-name lab iface enX type ethernet \
    ipv4.method manual \
    ipv4.address 172.25.250.11/24 ipv4.gateway 172.25.250.254
[root@serverb ~]# nmcli con mod "lab" ipv4.dns 172.25.250.254
```

- Configure the new connection to be autostarted. Other connections should not start automatically.

```
[root@serverb ~]# nmcli con mod "lab" connection.autoconnect yes
[root@serverb ~]# nmcli con mod "Wired connection 1" connection.autoconnect no
```

- Modify the new connection so that it also uses the address 10.0.1.1/24.

```
[root@serverb ~]# nmcli con mod "lab" +ipv4.addresses 10.0.1.1/24
```

Or alternately:

```
[root@serverb ~]# echo "IPADDR1=10.0.1.1" \
>> /etc/sysconfig/network-scripts/ifcfg-lab
[root@serverb ~]# echo "PREFIX1=24" >> /etc/sysconfig/network-scripts/ifcfg-lab
```

- Configure the **hosts** file so that 10.0.1.1 can be referenced as **private**.

```
[root@serverb ~]# echo "10.0.1.1 private" >> /etc/hosts
```

7. Reboot the system.

```
[root@serverb ~]# systemctl reboot
Connection to serverb closed by remote host.
Connection to serverb closed.
[student@workstation ~]$
```

8. From **workstation** use the **ping** command to verify that **serverb** is initialized.

```
[student@workstation ~]$ ping -c3 serverb
PING serverb.lab.example.com (172.25.250.11) 56(84) bytes of data.
64 bytes from serverb.lab.example.com (172.25.250.11): icmp_seq=1 ttl=64
time=0.478 ms
64 bytes from serverb.lab.example.com (172.25.250.11): icmp_seq=2 ttl=64
time=0.504 ms
64 bytes from serverb.lab.example.com (172.25.250.11): icmp_seq=3 ttl=64
time=0.513 ms

--- serverb.lab.example.com ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 78ms
rtt min/avg/max/mdev = 0.478/0.498/0.513/0.023 ms
[student@workstation ~]$
```

Evaluation

On workstation, run the **lab net-review grade** script to confirm success on this lab.

```
[student@workstation ~]$ lab net-review grade
```

Finish

On **workstation**, run the **lab net-review finish** script to finish this lab.

```
[student@workstation ~]$ lab net-review finish
```

This concludes the lab.

Summary

In this chapter, you learned:

- The TCP/IP network model is a simplified, four-layered set of abstractions that describes how different protocols interoperate in order for computers to send traffic from one machine to another over the Internet.
- IPv4 is the primary network protocol used on the Internet today. IPv6 is intended as an eventual replacement for the IPv4 network protocol. By default, Red Hat Enterprise Linux operates in dual-stack mode, using both protocols in parallel.
- NetworkManager is a daemon that monitors and manages network configuration.
- The **nmcli** command is a command-line tool for configuring network settings with NetworkManager.
- The system's static host name is stored in the **/etc/hostname** file. The **hostnamectl** command is used to modify or view the status of the system's host name and related settings. The **hostname** command displays or temporarily modifies the system's host name.

Archiving and Transferring Files

Goal

Archive and copy files from one system to another.

Objectives

- Archive files and directories into a compressed file using tar, and extract the contents of an existing tar archive.
- Transfer files to or from a remote system securely using SSH.
- Synchronize the contents of a local file or directory with a copy on a remote server.

Sections

- Managing Compressed Tar Archives (and Guided Exercise)
- Transferring Files Between Systems Securely (and Guided Exercise)
- Synchronizing Files Between Systems Securely (and Guided Exercise)

Lab

Archiving and Transferring Files

Managing Compressed tar Archives

Objectives

After completing this section, you should be able to archive files and directories into a compressed file using **tar**, and extract the contents of an existing **tar** archive.

The tar Command

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). A **tar** archive is a structured sequence of file data mixed in with metadata about each file and an index so that individual files can be extracted. 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.

Selected tar Options

tar command options are divided into operations (the action you want to take): general options and compression options. The table below shows common options, long version of options, and their description:

Overview of tar Operations

Option	Description
-c, --create	Create a new archive.
-x, --extract	Extract from an existing archive.
-t, --list	List the table of contents of an archive.

Selected tar General Options

Option	Description
-v, --verbose	Verbose. Shows which files get archived or extracted.
-f, --file=	File name. This option must be followed by the file name of the archive to use or create.
-p, --preserve-permissions	Preserve the permissions of files and directories when extracting an archive, without subtracting the umask .

Overview of tar Compression Options

Option	Description
-z, --gzip	Use gzip compression (.tar.gz).
-j, --bzip2	Use bzip2 compression (.tar.bz2). bzip2 typically achieves a better compression ratio than gzip.
-J, --xz	Use xz compression (.tar.xz). The xz compression typically achieves a better compression ratio than bzip2.

Listing Options of the tar Command

The **tar** command expects one of the three following options:

- Use the **-c** or **--create** option to create an archive.
- Use the **-t** or **--list** option to list the contents of an archive.
- Use the **-x** or **--extract** option to extract an archive.

Other commonly used options are:

- Use the **-f** or **--file=** option with a file name as an argument of the archive to operate.
- Use the **-v** or **--verbose** option for verbosity; useful to see which files get added to or extracted from the archive.



Note

The **tar** command actually supports a third, old option style that uses the standard single-letter options with no leading **-**. It is still commonly encountered, and you might run into this syntax when working with other people's instructions or commands. The **info tar 'old options'** command discusses how this differs from normal short options in some detail.

You can ignore old options for now and focus on the standard short and long options syntax.

Archiving Files and Directories

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.



Warning

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 overwrites an existing archive without warning.

The following command creates an archive named **archive.tar** 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
```

The above **tar** command can also be executed using the long version options.

```
[user@host ~]$ tar --file=archive.tar --create file1 file2 file3
```



Note

When archiving files by absolute path names, the leading `/` of the path is removed from the file name by default. Removing the leading `/` of the path help users to avoid overwriting important files when extracting the archive. The **tar** command extracts files relative to the current working directory.

For tar to be able to archive the selected files, it is mandatory that the user executing the **tar** command can read the files. For example, creating a new archive of the `/etc` folder and all of its content requires **root** privileges, because only the **root** user is allowed to read all of the files present in the `/etc` directory. An unprivileged user can create an archive of the `/etc` directory, but the archive omits files which do not include read permission for the user, and it omits directories which do not include both read and execute permission for the user.

To create the tar archive named, `/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

Some advanced permissions that we have not covered in this course, such as ACLs and SELinux contexts, are not automatically stored in a **tar** archive. Use the `--xattrs` option when creating an archive to store those extended attributes in the tar archive.

Listing Contents of an Archive

The **t** option directs **tar** to list the contents (table of contents, hence **t**) of the archive. Use the **f** option with the name of the archive to be queried. For example:

```
[root@host ~]# tar -tf /root/etc.tar
etc/
etc/fstab
etc/crypttab
etc/mtab
...output omitted...
```

Extracting Files from an Archive

A tar archive should usually be extracted in an empty directory to ensure it does not overwrite any existing files. When **root** extracts an archive, the **tar** command preserves the original user and group ownership of the files. If a regular user extracts files using **tar**, the file ownership belongs to the user extracting the files from the archive.

To restore files from the **/root/etc.tar** archive to the **/root/etcbackup** directory, run:

```
[root@host ~]# mkdir /root/etcbackup
[root@host ~]# cd /root/etcbackup
[root@host etcbackup]# tar -tf /root/etc.tar
etc/
etc/fstab
etc/crypttab
etc/mtab
...output omitted...
[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. To preserve the permissions of an archived file, the **p** option when extracting an archive.

In this example, an archive named, **/root/myscripts.tar**, is extracted in 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
```

Creating a Compressed Archive

The **tar** command supports three compression methods. 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 across distributions and even across platforms. **bzip2** compression creates smaller archive files compared to **gzip** but 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 type of data that is 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 the extraction of the files in an organized way.

Use one of the following options to create a compressed tar archive:

- **-z** or **--gzip** for **gzip** compression (**filename.tar.gz** or **filename.tgz**)
- **-j** or **--bzip2** for **bzip2** compression (**filename.tar.bz2**)
- **-J** or **-xz** for **xz** compression (**filename.tar.xz**)

To create a **gzip** compressed archive named **/root/etcbackup.tar.gz**, with the contents from the **/etc** directory on **host**:

```
[root@host ~]# tar -czf /root/etcbackup.tar.gz /etc  
tar: Removing leading `/' from member names
```

To create a **bzip2** compressed archive named **/root/logbackup.tar.bz2**, with the contents from the **/var/log** directory on **host**:

```
[root@host ~]$ tar -cjf /root/logbackup.tar.bz2 /var/log  
tar: Removing leading `/' from member names
```

To create a **xz** compressed archive named, **/root/sshconfig.tar.xz**, with the contents from the **/etc/ssh** directory on **host**:

```
[root@host ~]$ tar -cJf /root/sshconfig.tar.xz /etc/ssh  
tar: Removing leading `/' from member names
```

After creating an archive, verify the content of an archive using the **tf** options. It is not mandatory to use the option for compression agent when listing the content of a compressed archive file. For example, to list the content archived in the **/root/etcbackup.tar.gz** file, which uses the **gzip** compression, use the following command:

```
[root@host ~]# tar -tf /root/etcbackup.tar.gz /etc  
etc/  
etc/fstab  
etc/crypttab  
etc/mtab  
...output omitted...
```

Extracting a Compressed 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. The **tar** command determines which compression was used and it is usually not necessary to use the same compression option used when creating the archive. It is valid to add the decompression method to the **tar** command. If one chooses to do so, the correct decompression type option must be used; otherwise tar yields an error about the decompression type specified in the options not matching the file's decompression type.

To extract the contents of a **gzip** compressed archive named **/root/etcbackup.tar.gz** in the **/tmp/etcbackup** directory:

```
[root@host ~]# mkdir /tmp/etcbackup  
[root@host ~]# cd /tmp/etcbackup  
[root@host etcbackup]# tar -tf /root/etcbackup.tar.gz  
etc/  
etc/fstab  
etc/crypttab
```

```
etc/mtab
...output omitted...
[root@host etcbackup]# tar -xzf /root/etcbackup.tar.gz
```

To extract the contents of a **bzip2** compressed archive named **/root/logbackup.tar.bz2** in the **/tmp/logbackup** directory:

```
[root@host ~]# mkdir /tmp/logbackup
[root@host ~]# cd /tmp/logbackup
[root@host logbackup]# tar -tf /root/logbackup.tar.bz2
var/log/
var/log/lastlog
var/log/README
var/log/private/
var/log/wtmp
var/log/btmp
...output omitted...
[root@host logbackup]# tar -xjf /root/logbackup.tar.bz2
```

To extract the contents of a **xz** compressed archive named **/root/sshbackup.tar.xz** in the **/tmp/sshbackup** directory:

```
[root@host ~]$ mkdir /tmp/sshbackup
[root@host ~]# cd /tmp/sshbackup
[root@host sshbackup]# tar -tf /root/sshbackup.tar.xz
etc/ssh/
etc/ssh/moduli
etc/ssh/ssh_config
etc/ssh/ssh_config.d/
etc/ssh/ssh_config.d/05-redhat.conf
etc/ssh/sshd_config
...output omitted...
[root@host sshbackup]# tar -xJf /root/sshbackup.tar.xz
```

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, the **gzip etc.tar** command results in the **etc.tar.gz** compressed file, while the **bzip2 abc.tar** command results in the **abc.tar.bz2** compressed file, and the **xz myarchive.tar** command results in the **myarchive.tar.xz** compressed file.

The corresponding commands to decompress are **gunzip**, **bunzip2**, and **unxz**. For example, the **gunzip /tmp/etc.tar.gz** command results in the **etc.tar** uncompressed tar file, while the **bunzip2 abc.tar.bz2** command results in the **abc.tar** uncompressed tar file, and the **unxz myarchive.tar.xz** command results in the **myarchive.tar** uncompressed tar file.



References

tar(1), gzip(1), gunzip(1), bzip2(1), bunzip2(1), xz(1), unxz(1) man pages

► Guided Exercise

Managing Compressed Tar Archives

In this exercise, you will create archive files and extract their contents with the tar command.

Outcomes

You should be able to archive a directory tree and extract the archive content to another location.

Before You Begin

Log in as the **student** user on **workstation** using **student** as the password.

From **workstation**, run the **lab archive-manage start** command. The command runs a start script that determines if the host, **servera**, is reachable on the network. The script also ensures that the file and directory to be created in the exercise does not exist on **servera**.

```
[student@workstation ~]$ lab archive-manage start
```

- 1. Use the **ssh** command to log in to **servera** as the **student** user.

```
[student@workstation ~]$ ssh student@servera  
...output omitted...  
[student@servera ~]$
```

- 2. Switch to the **root** user as only the **root** user can access all of the content of the **/etc** directory.

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

- 3. Use the **tar** command with the **-czf** options to create an archive of the **/etc** directory using **gzip** compression. Save the archive file as **/tmp/etc.tar.gz**.

```
[root@servera ~]# tar -czf /tmp/etc.tar.gz /etc  
tar: Removing leading `/' from member names  
[root@servera ~]#
```

- 4. Use the **tar** command with the **-tzf** options to verify that the **etc.tar.gz** archive contains the files from the **/etc** directory.

```
[root@servera ~]# tar -tzf /tmp/etc.tar.gz
etc/
etc/mtab
etc/fstab
etc/crypttab
etc/resolv.conf
...output omitted...
```

- 5. On **servera**, create a directory named **/backuptest**. Verify that the **etc.tar.gz** backup file is a valid archive by decompressing the file to the **/backuptest** directory.

- 5.1. Create the **/backuptest** directory.

```
[root@servera ~]# mkdir /backuptest
```

- 5.2. Change to the **/backuptest** directory.

```
[root@servera ~]# cd /backuptest
[root@servera backuptest]#
```

- 5.3. List the contents of the **etc.tar.gz** archive before extracting.

```
[root@servera backuptest]# tar -tzf /tmp/etc.tar.gz
etc/
etc/mtab
etc/fstab
etc/crypttab
etc/resolv.conf
...output omitted...
```

- 5.4. Extract the **/tmp/etc.tar.gz** archive to the **/backuptest** directory.

```
[root@servera backuptest]# tar -xzf /tmp/etc.tar.gz
[root@servera backuptest]#
```

- 5.5. List the content of the **/backuptest** directory. Verify that the directory contains the files from the **/etc** directory.

```
[root@servera backuptest]# ls -l
total 12
drwxr-xr-x. 95 root root 8192 Feb  8 10:16 etc
[root@servera backuptest]# cd etc
[root@servera etc]# ls -l
total 1204
-rw-r--r--.  1 root root      16 Jan 16 23:41 adjtime
-rw-r--r--.  1 root root    1518 Sep 10 17:21 aliases
drwxr-xr-x.  2 root root     169 Feb  4 21:58 alternatives
-rw-r--r--.  1 root root     541 Oct  2 21:01 anacrontab
...output omitted...
```

► **6.** Exit from **servera**.

```
[root@servera backuptest]# exit  
logout  
[student@servera ~]$ exit  
logout  
Connection to servera closed.  
[student@workstation]$
```

Finish

On **workstation**, run the **lab archive-manage finish** script to complete this exercise.

```
[student@workstation ~]$ lab archive-manage finish
```

This concludes the guided exercise.

Transferring Files Between Systems Securely

Objectives

After completing this section, you should be able to transfer files to or from a remote system securely using SSH.

Transferring Files Using Secure Copy

OpenSSH is useful for securely running shell commands on remote systems. The Secure Copy command, **scp**, which is part of the OpenSSH suite, copies files from a remote system to the local system or from the local system to a remote system. The command uses the SSH server for authentication and encrypts data when it is being transferred.

You can specify a remote location for the source or destination of the files you are copying. The format of the remote location should be in the form **[user@]host :/path**. The **user@** portion of the argument is optional. If it is missing, your current local username will be used. When you run the command, your **scp** client will authenticate to the remote SSH server just like **ssh**, using key-based authentication or prompting you for your password.

The following example demonstrates how to copy the local **/etc/yum.conf** and **/etc/hosts** files on **host**, to the **remoteuser**'s home directory on the **remotehost** remote system:

```
[user@host ~]$ scp /etc/yum.conf /etc/hosts remoteuser@remotehost:/home/remoteuser
remoteuser@remotehost's password: password
      yum.conf                                100%   813      0.8KB/s  00:00
      hosts                                    100%   227      0.2KB/s  00:00
```

You can also copy a file in the other direction, from a remote system to the local file system. In this example, the file **/etc/hostname** on **remotehost** is copied to the local directory **/home/user**. The **scp** command authenticates to **remotehost** as the user **remoteuser**.

```
[user@host ~]$ scp remoteuser@remotehost:/etc/hostname /home/user
remoteuser@remotehost's password: password
      hostname                               100%    22      0.0KB/s  00:00
```

To copy a whole directory tree recursively, use the **-r** option. In the following example, the remote directory **/var/log** on **remotehost** is copied recursively to the local directory **/tmp/** on **host**. You must connect to the remote system as **root** to make sure you can read all files in the remote **/var/log** directory.

```
[user@host ~]$ scp -r root@remoteuser:/var/log /tmp
root@remotehost's password: password
...output omitted...
```

Transferring Files Using the Secure File Transfer Program

To interactively upload or download files from a SSH server, use the Secure File Transfer Program, **sftp**. A session with the **sftp** command uses the secure authentication mechanism and encrypted data transfer to and from the SSH server.

Just like the **scp** command, the **sftp** command uses **[user@]host** to identify the target system and user name. If you do not specify a user, the command will attempt to log in using your local user name as the remote user name. You will then be presented with an **sftp>** prompt.

```
[user@host ~]$ sftp remoteuser@remotehost
remoteuser@remotehost's password: password
Connected to remotehost.
sftp>
```

The interactive **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**. The **put** command uploads a file to the remote system. The **get** command downloads a file from the remote system. The **exit** command exits the **sftp** session.

To upload the **/etc/hosts** file on the local system to the newly created directory **/home/remoteuser/hostbackup** on **remotehost**. 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/remoteuser**:

```
sftp> mkdir hostbackup
sftp> cd hostbackup
sftp> put /etc/hosts
Uploading /etc/hosts to /home/remoteuser/hostbackup/hosts
/etc/hosts                                         100%   227      0.2KB/s  00:00
sftp>
```

To download **/etc/yum.conf** from the remote host to the current directory on the local 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
[user@host ~]$
```



References

scp(1) and **sftp(1)** man pages

► Guided Exercise

Transferring Files Between Systems Securely

In this exercise, you will copy files from a remote system to a local directory using **scp**.

Outcomes

You should be able to copy files from a remote host to a directory on the local machine.

Before You Begin

Log in as the **student** user on **workstation** using **student** as the password.

From **workstation**, run the **lab archive-transfer start** command. The command runs a start script that determines whether hosts **servera** and **serverb** are reachable on the network. The script also ensures that the file and directory to be created in the exercise does not exist on **servera**.

```
[student@workstation ~]$ lab archive-transfer start
```

- 1. Use the **ssh** command to log in to **servera** as the **student** user.

```
[student@workstation ~]$ ssh student@servera
...output omitted...
[student@servera ~]$
```

- 2. Use the **scp** command to copy the **/etc/ssh** directory from **serverb** to the **/home/student/serverbackup** directory on **servera**.

- 2.1. On **servera**, create a directory named **/home/student/serverbackup**.

```
[student@servera ~]$ mkdir ~/serverbackup
```

- 2.2. Use the **scp** command to recursively copy the **/etc/ssh** directory from **serverb** to the **/home/student/serverbackup** directory on **servera**. When prompted, enter **redhat** as the password. Note that only the **root** user can read all the content in the **/etc/ssh** directory.

```
[student@servera ~]$ scp -r root@serverb:/etc/ssh ~/serverbackup
The authenticity of host 'serverb (172.25.250.11)' can't be established.
ECDSA key fingerprint is SHA256:qaS0PToLrqIc02XGkIA0iY7CaP7aPKimerDoaUkv720.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added 'serverb,172.25.250.11' (ECDSA) to the list of known
hosts.
root@serverb's password: redhat
moduli                                         100%   550KB  57.9MB/s   00:00
```

ssh_config	100%	1727	1.4MB/s	00:00
05-redhat.conf	100%	690	1.6MB/s	00:00
01-training.conf	100%	36	80.5KB/s	00:00
ssh_host_ed25519_key	100%	387	1.2MB/s	00:00
ssh_host_ed25519_key.pub	100%	82	268.1KB/s	00:00
ssh_host_ecdsa_key	100%	492	1.5MB/s	00:00
ssh_host_ecdsa_key.pub	100%	162	538.7KB/s	00:00
ssh_host_rsa_key	100%	1799	4.9MB/s	00:00
ssh_host_rsa_key.pub	100%	382	1.2MB/s	00:00
sshd_config	100%	4469	9.5MB/s	00:00

- 2.3. Verify that the **/etc/ssh** directory from **serverb** is copied to the **/home/student/serverbackup** directory to **servera**.

```
[student@servera ~]$ ls -lR ~/serverbackup
/home/student/serverbackup:
total 0
drwxr-xr-x. 3 student student 245 Feb 11 18:35 ssh

/home/student/serverbackup/ssh:
total 588
-rw-r--r--. 1 student student 563386 Feb 11 18:35 moduli
-rw-r--r--. 1 student student 1727 Feb 11 18:35 ssh_config
drwxr-xr-x. 2 student student 28 Feb 11 18:35 ssh_config.d
-rw-----. 1 student student 4469 Feb 11 18:35 sshd_config
-rw-r-----. 1 student student 492 Feb 11 18:35 ssh_host_ecdsa_key
-rw-r--r--. 1 student student 162 Feb 11 18:35 ssh_host_ecdsa_key.pub
-rw-r-----. 1 student student 387 Feb 11 18:35 ssh_host_ed25519_key
-rw-r--r--. 1 student student 82 Feb 11 18:35 ssh_host_ed25519_key.pub
-rw-r-----. 1 student student 1799 Feb 11 18:35 ssh_host_rsa_key
-rw-r--r--. 1 student student 382 Feb 11 18:35 ssh_host_rsa_key.pub

/home/student/serverbackup/ssh/sshd_config.d:
total 8
-rw-r--r--. 1 student student 36 Feb 11 18:35 01-training.conf
-rw-r--r--. 1 student student 690 Feb 11 18:35 05-redhat.conf
```

► 3. Exit from **servera**.

```
[student@servera ~]$ exit
logout
Connection to servera closed.
[student@workstation]$
```

Finish

On **workstation**, run the **lab archive-transfer finish** script to complete this exercise.

```
[student@workstation ~]$ lab archive-transfer finish
```

This concludes the guided exercise.

Synchronizing Files Between Systems Securely

Objectives

After completing this section, you should be able to efficiently and securely synchronize the contents of a local file or directory with a copy on a remote server.

Synchronize Files and Directories with rsync

The **rsync** command is another way to securely copy files from one system to another. The tool uses an algorithm that minimizes the amount of data copied by synchronizing only the changed portions of files. It differs from **scp** in that if two files or directories are similar between two servers, **rsync** copies only the differences between the file systems, while **scp** would still copy everything.

An advantage of **rsync** is that it can copy files between a local system and a remote system securely and efficiently. While an initial directory synchronization takes about the same time as copying it, subsequent synchronizations only require the differences to be copied over the network, substantially speeding updates.

An important option of **rsync** is the **-n** option to perform a dry run. A dry run is a simulation of what happens when the command gets executed. The dry run shows the changes **rsync** would perform when the command is run normally. Perform a dry run before the actual **rsync** operation to ensure no important files get overwritten or deleted.

Two common options when synchronizing with **rsync** are the **-v** and **-a** options.

The **-v** or **--verbose** option provides more detailed output. This is useful for troubleshooting and to view live progress.

The **-a** or **--archive** option enables "archive mode". This enables recursive copying and turns on a large number of useful options that preserve most characteristics of the files. Archive mode is the same as specifying the following options:

Options Enabled with rsync -a (Archive Mode)

Option	Description
-r, --recursive	synchronize recursively the whole directory tree
-l, --links	synchronize symbolic links
-p, --perms	preserve permissions
-t, --times	preserve time stamps
-g, --group	preserve group ownership
-o, --owner	preserve the owner of the files
-D, --devices	synchronize device file

Archive mode does not preserve hard links, because this can add significant time to the synchronization. If you want to preserve hard links too, add the **-H** option.

**Note**

If you are using advanced permissions, you might need two additional options:

- **-A** to preserve ACLs
- **-X** to preserve SELinux contexts

You can use **rsync** to synchronize the contents of a local file or directory with a file or directory on a remote machine, using either machine as the source. You can also synchronize the contents of two local files or directories.

For example, to synchronize contents of the **/var/log** directory to the **/tmp** directory:

```
[user@host ~]$ su -
Password: password
[root@host ~]# rsync -av /var/log /tmp
receiving incremental file list
log/
log/README
log/boot.log
...output omitted...
log/tuned/tuned.log

sent 11,592,423 bytes received 779 bytes 23,186,404.00 bytes/sec
total size is 11,586,755 speedup is 1.00
[root@host ~]$ ls /tmp
log ssh-RLjDdarkKiW1
[root@host ~]$
```

A trailing slash on the source directory synchronizes the content of that directory without newly creating the subdirectory in the target directory. In this example, the **log** directory is *not* created in the **/tmp** directory, only the content of **/var/log/** is synchronized into **/tmp**.

```
[root@host ~]# rsync -av /var/log/ /tmp
sending incremental file list
./
README
boot.log
...output omitted...
tuned/tuned.log

sent 11,592,389 bytes received 778 bytes 23,186,334.00 bytes/sec
total size is 11,586,755 speedup is 1.00
[root@host ~]# ls /tmp
anaconda          dnf.rpm.log-20190318  private
audit             dnf.rpm.log-20190324  qemu-ga
boot.log          dnf.rpm.log-20190331  README
...output omitted...
```

 **Important**

When typing the source directory in the **rsync** command, it is significant whether a trailing slash is present on the directory name. It determines whether the *directory* or just the *contents of the directory* are synchronized to the target.

Bash **Tab**-completion automatically adds a trailing slash to directory names.

Like the **scp** and **sftp** commands, **rsync** specifies remote locations using the **[user@]host:/path** format. The remote location can be either the source or destination system, but one of the two machines has to be local.

To preserve file ownership, you need to be **root** on the destination system. If the destination is remote, authenticate as **root**. If the destination is local, you must run **rsync** as **root**.

In this example, synchronize the local **/var/log** directory to the **/tmp** directory on the **remotehost** system:

```
[root@host ~]# rsync -av /var/log remotehost:/tmp
root@remotehost's password: password
receiving incremental file list
log/
log/README
log/boot.log
...output omitted...
sent 9,783 bytes received 290,576 bytes 85,816.86 bytes/sec
total size is 11,585,690 speedup is 38.57
```

In the same way, the **/var/log** remote directory on **remotehost** can be synchronized to the **/tmp** local directory on **host**:

```
[root@host ~]# rsync -av remotehost:/var/log /tmp
root@remotehost's password: password
receiving incremental file list
log/boot.log
log/dnf.librepo.log
log/dnf.log
...output omitted...

sent 9,783 bytes received 290,576 bytes 85,816.86 bytes/sec
total size is 11,585,690 speedup is 38.57
```


References

[rsync\(1\) man page](#)

► Guided Exercise

Synchronizing Files Between Systems Securely

In this exercise, you will synchronize the contents of a local directory with a copy on a remote server by using **rsync** over SSH.

Outcomes

You should be to use the **rsync** command to synchronize the content of a local directory with a copy on a remote server.

Before You Begin

Log in as the **student** user on **workstation** using **student** as the password.

From **workstation**, run the **lab archive-sync start** command. The command runs a start script that determines if the hosts, **servera** and **serverb**, are reachable on the network. The script also ensures that the file and directory to be created in the exercise does not exist on **servera**.

```
[student@workstation ~]$ lab archive-sync start
```

- 1. Use the **ssh** command to log in to **servera** as the **student** user.

```
[student@workstation ~]$ ssh student@servera
...output omitted...
[student@servera ~]$
```

- 2. Create a directory named **/home/student/serverlogs** on **servera**. Use the **rsync** command to securely create an initial copy of the **/var/log** directory tree on **serverb** in the **/home/student/serverlogs** directory on **servera**.
 - 2.1. On **servera**, create the target directory named **/home/student/serverlogs** to store the log files synchronized from **serverb**.

```
[student@servera ~]$ mkdir ~/serverlogs
```

- 2.2. Use the **rsync** command to synchronize the **/var/log** directory tree on **serverb** to the **/home/student/serverlogs** directory on **servera**. Note that only the **root** user can read all the content in the **/var/log** directory on **serverb**. All files are transferred in the initial synchronization.

```
[student@servera ~]$ rsync -av root@serverb:/var/log ~/serverlogs
root@serverb's password: redhat
receiving incremental file list
log/
```

```
log/README
log/boot.log
...output omitted...
log/tuned/tuned.log

sent 992 bytes received 13,775,064 bytes 2,119,393.23 bytes/sec
total size is 13,768,109 speedup is 1.00
```

- 3. As **root** on **serverb**, execute the **logger "Log files synchronized"** command to get a new entry in the **/var/log/messages** log file to reflect when the last synchronization took place.

```
[student@servera ~]$ ssh root@serverb 'logger "Log files synchronized"'
Password: redhat
[student@servera ~]$
```

- 4. Use the **rsync** command to securely synchronize from the **/var/log** directory tree on **serverb** to the **/home/student/serverlogs** directory on **servera**. Note that this time only log files that have changed are transferred.

```
[student@servera ~]$ rsync -av root@serverb:/var/log ~/serverlogs
root@serverb's password: redhat
receiving incremental file list
log/messages
log/secure
log/audit/audit.log

sent 3,496 bytes received 27,243 bytes 8,782.57 bytes/sec
total size is 11,502,695 speedup is 374.21
```

- 5. Exit from **servera**.

```
[student@servera ~]$ exit
logout
Connection to servera closed.
[student@workstation]$
```

Finish

On **workstation**, run the **lab archive-sync finish** script to complete this exercise.

```
[student@workstation ~]$ lab archive-sync finish
```

This concludes the guided exercise.

▶ Lab

Archiving and Transferring Files

Performance Checklist

In this lab, you will use **tar**, **rsync**, and **scp** to archive and back up the contents of directories.

Outcomes

You should be able to:

- Synchronize a remote directory to a local directory.
- Create an archive of the contents of a synchronized directory.
- Securely copy an archive to a remote host.
- Extract an archive.

Before You Begin

Log in as the **student** user on **workstation** using **student** as the password.

From **workstation**, run the **lab archive-review start** command. The command runs a start script that determines whether hosts **servera** and **serverb** are reachable on the network. The script also ensures that the files and directories to be created in the lab do not exist on **serverb**, and **workstation**.

```
[student@workstation ~]$ lab archive-review start
```

1. On **serverb**, synchronize the **/etc** directory tree from **servera** to the **/configsync** directory.
2. Use **gzip** compression to create an archive named **configfile-backup-servera.tar.gz** with the contents of the **/configsync** directory.
3. Securely copy the **/root/configfile-backup-servera.tar.gz** archive file from **serverb** to the **/home/student** directory on **workstation** as the **student** user using **student** as the password.
4. On **workstation**, extract the contents of the **/home/student/configfile-backup-servera.tar.gz** archive to the **/tmp/savedconfig/** directory.
5. On **workstation** return to the **student** home directory.

```
[student@workstation savedconfig]$ cd
```

Evaluation

On **workstation**, run the **lab archive-review grade** script to confirm success on this lab.

```
[student@workstation ~]$ lab archive-review grade
```

Finish

On **workstation**, run the **lab archive-review finish** script to complete this exercise.

```
[student@workstation ~]$ lab archive-review finish
```

This concludes the lab.

► Solution

Archiving and Transferring Files

Performance Checklist

In this lab, you will use **tar**, **rsync**, and **scp** to archive and back up the contents of directories.

Outcomes

You should be able to:

- Synchronize a remote directory to a local directory.
- Create an archive of the contents of a synchronized directory.
- Securely copy an archive to a remote host.
- Extract an archive.

Before You Begin

Log in as the **student** user on **workstation** using **student** as the password.

From **workstation**, run the **lab archive-review start** command. The command runs a start script that determines whether hosts **servera** and **serverb** are reachable on the network. The script also ensures that the files and directories to be created in the lab do not exist on **serverb**, and **workstation**.

```
[student@workstation ~]$ lab archive-review start
```

1. On **serverb**, synchronize the **/etc** directory tree from **servera** to the **/configsync** directory.

- 1.1. Use the **ssh** command to log in to **serverb** as the **student** user.

```
[student@workstation ~]$ ssh student@serverb
...output omitted...
[student@serverb ~]$
```

- 1.2. Use the **su** command to switch to the **root** user because creating the **/configsync** directory requires superuser privileges. In later steps, you will archive the files present in the **/etc** directory tree, which are owned by the **root** user; this also requires superuser privileges.

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

- 1.3. Create the **/configsync** directory to store the synchronized configuration files from **servera**.

```
[root@serverb ~]# mkdir /configsync
```

- 1.4. Use the **rsync** command to synchronize the **/etc** directory tree from **servera** to the **/configsync** directory on **serverb**.

Be aware that only the **root** user can read all the content in the **/etc** directory on **servera**.

```
[root@serverb ~]# rsync -av root@servera:/etc /configsync
root@servera's password: redhat
receiving incremental file list
etc/
etc/.pwd.lock
...output omitted...
etc/yum/protected.d -> ../dnf/protected.d
etc/yum/vars -> ../dnf/vars

sent 10,958 bytes received 21,665,987 bytes 3,334,914.62 bytes/sec
total size is 21,615,767 speedup is 1.00
```

2. Use **gzip** compression to create an archive named **configfile-backup-servera.tar.gz** with the contents of the **/configsync** directory.

- 2.1. Use the **tar** command with the **-czf** options to create a **gzip** compressed archive.

```
[root@serverb ~]# tar -czf configfile-backup-servera.tar.gz /configsync
tar: Removing leading `/' from member names
[root@serverb ~]#
```

- 2.2. Use the **tar** command with the **-tzf** options to list the contents of the **configfile-backup-servera.tar.gz** archive.

```
[root@serverb ~]# tar -tzf configfile-backup-servera.tar.gz
...output omitted...
configsync/etc/vimrc
configsync/etc/wgetrc
configsync/etc/xattr.conf
```

3. Securely copy the **/root/configfile-backup-servera.tar.gz** archive file from **serverb** to the **/home/student** directory on **workstation** as the **student** user using **student** as the password.

```
[root@serverb ~]# scp ~/configfile-backup-servera.tar.gz \
student@workstation:/home/student
...output omitted...
student@workstation's password: student
configfile-backup-servera.tar.gz          100% 5110KB 64.5MB/s 00:00
```

4. On **workstation**, extract the contents of the **/home/student/configfile-backup-servera.tar.gz** archive to the **/tmp/savedconfig/** directory.

- 4.1. Exit from **serverb**.

```
[root@serverb ~]# exit
logout
[student@serverb ~]$ exit
logout
Connection to serverb closed.
[student@workstation]$
```

- 4.2. Create the **/tmp/savedconfig** directory, where the contents of the **/home/student/configfile-backup-servera.tar.gz** archive will be extracted.

```
[student@workstation ~]$ mkdir /tmp/savedconfig
```

- 4.3. Change to the **/tmp/savedconfig** directory.

```
[student@workstation ~]$ cd /tmp/savedconfig
[student@workstation savedconfig]$
```

- 4.4. Use the **tar** command with the **-tzf** options to list the contents of the **configfile-backup-servera.tar.gz** archive

```
[student@workstation savedconfig]$ tar -tzf ~/configfile-backup-servera.tar.gz
...output omitted...
configsync/etc/vimrc
configsync/etc/wgetrc
configsync/etc/xattr.conf
```

- 4.5. Use the **tar** command with the **-xzf** options to extract the contents of the **/home/student/configfile-backup-servera.tar.gz** archive to the **/tmp/savedconfig/** directory.

```
[student@workstation savedconfig]$ tar -xzf ~/configfile-backup-servera.tar.gz
[student@workstation savedconfig]$
```

- 4.6. List the directory tree to verify that the directory contains files from the **/etc** directory.

```
[student@workstation savedconfig]$ ls -lR
.:
total 0
drwxr-xr-x. 3 student student 17 Feb 13 10:13 configsync

./configsync:
total 12
drwxr-xr-x. 95 student student 8192 Feb 13 09:41 etc

./configsync/etc:
total 1212
-rw-r--r--. 1 student student 16 Jan 16 23:41 adjtime
```

```
-rw-r--r--. 1 student student 1518 Sep 10 17:21 aliases  
drwxr-xr-x. 2 student student 169 Feb  4 21:58 alternatives  
...output omitted...
```

5. On **workstation** return to the **student** home directory.

```
[student@workstation savedconfig]$ cd
```

Evaluation

On **workstation**, run the **lab archive-review grade** script to confirm success on this lab.

```
[student@workstation ~]$ lab archive-review grade
```

Finish

On **workstation**, run the **lab archive-review finish** script to complete this exercise.

```
[student@workstation ~]$ lab archive-review finish
```

This concludes the lab.

Summary

In this chapter, you learned:

- The **tar** command creates an archive file from a set of files and directories, extracts files from the archive, and lists the contents of an archive.
- The **tar** command provides a set of different compression methods reduce archive size.
- Besides providing a secure remote shell, the **SSH** service also provides the **scp** and **sftp** commands as secure ways to transfer files from and to a remote system running the **SSH** server.
- The **rsync** command securely and efficiently synchronizes files between two directories, either one of which can be on a remote system.

Installing and Updating Software Packages

Goal

Download, install, update, and manage software packages from Red Hat and Yum package repositories.

Objectives

- Register a system to your Red Hat account and assign it entitlements for software updates and support services using Red Hat Subscription Management.
- Explain how software is provided as RPM packages, and investigate the packages installed on the system with Yum and RPM.
- Find, install, and update software packages using the **yum** command.
- Enable and disable use of Red Hat or third-party Yum repositories by a server.
- Explain how modules allow installation of specific versions of software, list, enable, and switch module streams, and install and update packages from a module.

Sections

- Registering Systems for Red Hat Support (and Quiz)
- Explaining and Investigating RPM Software Packages (and Quiz)
- Installing and Updating Software Packages with Yum (and Guided Exercise)
- Enabling Yum Software Repositories (and Guided Exercise)
- Managing Package Module Streams (and Guided Exercise)

Lab

Installing and Updating Software Packages

Registering Systems for Red Hat Support

Objectives

After completing this section, you should be able to register a system to your Red Hat account and assign it entitlements for software updates and support services using Red Hat Subscription Management.

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 that 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)*.

Registering a System

There are a number of different ways to register a system with Red Hat Customer Portal. There is a graphical interface that you can access with a GNOME application or through the Web Console service, and there is a command-line tool.

To register a system with the GNOME application, launch Red Hat Subscription Manager by selecting **Activities**. Type *subscription* in the **Type to search...** field and click on **Red Hat Subscription Manager**. Enter the appropriate password when prompted to authenticate. This displays the following **Subscriptions** window:

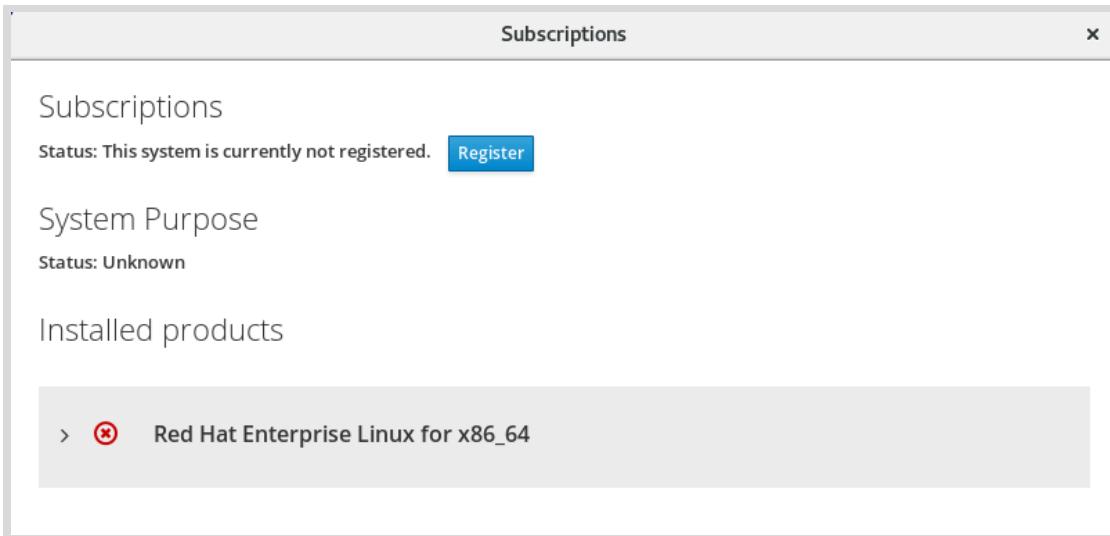


Figure 14.1: The main window of Red Hat Subscription Manager

To register the system, click the **Register** button in the **Subscriptions** window. This displays the following dialog:

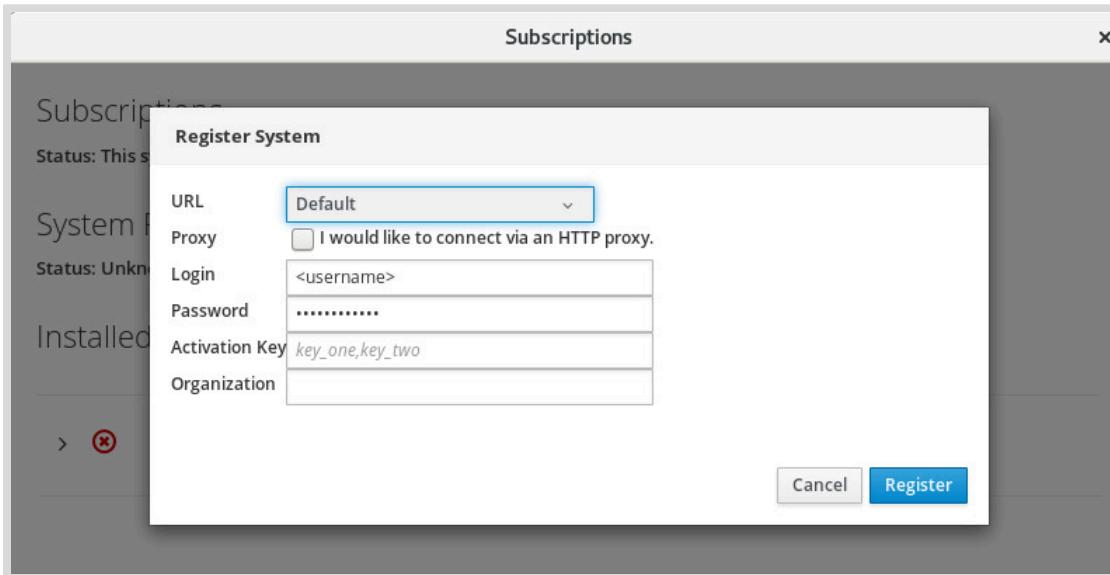


Figure 14.2: The service location and account information dialog of Red Hat Subscription Manager

This dialog box registers a system with a subscription server. By default, it registers the server to Red Hat Customer Portal. Provide the **Login** and **Password** for the Red Hat Customer Portal account to which the system should be registered, and click the **Register** button.

When registered, the system automatically has a subscription attached if one is available.

After the system is registered and a subscription has been assigned, close the **Subscriptions** window. The system is now properly subscribed and ready to receive updates or install new software from Red Hat.

Registration from the Command Line

Use the **subscription-manager(8)** command 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:

```
[user@host ~]$ subscription-manager register --username=yourusername \
--password=yourpassword
```

- View available subscriptions:

```
[user@host ~]$ subscription-manager list --available | less
```

- Auto-attach a subscription:

```
[user@host ~]$ subscription-manager attach --auto
```

- Alternatively, attach a subscription from a specific pool from the list of available subscriptions:

```
[user@host ~]$ subscription-manager attach --pool=poolID
```

- View consumed subscriptions:

```
[user@host ~]$ subscription-manager list --consumed
```

- Unregister a system:

```
[user@host ~]$ 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 often issued by an on-premise subscription management service, such as Subscription Asset Manager or Red Hat Satellite, and are not discussed in detail in this course.

Entitlement certificates

An entitlement is a subscription that has been attached to a system. Digital certificates are used to store current information about entitlements on the local system. Once registered, entitlement certificates are stored in **/etc/pki** and its subdirectories.

- **/etc/pki/product** contains certificates indicating which Red Hat products are installed on the system.
- **/etc/pki/consumer** contains certificates identifying the Red Hat account to which the system is registered.

- **/etc/pki/entitlement** contains certificates indicating which subscriptions are attached to the system.

The certificates can be inspected with the **rct** utility directly, but the **subscription-manager** tools provide easier ways to examine the subscriptions attached to the system.



References

subscription-manager(8) and **rct(8)** man pages

Get started with Red Hat Subscription Management

<https://access.redhat.com/site/articles/433903>

► Quiz

Registering Systems for Red Hat Support

Choose the correct answer to the following questions:

- ▶ 1. Which command is used to register a system without using a graphical environment?
 - a. **rct**
 - b. **subscription-manager**
 - c. **rpm**
 - d. **yum**
- ▶ 2. Which GUI tool is used to register and subscribe a system?
 - a. PackageKit
 - b. **gpk-application**
 - c. Red Hat Subscription Manager
 - d. **gnome-software**
- ▶ 3. Which task(s) can be performed with Red Hat Subscription Management tools?
 - a. Register a system.
 - b. Subscribe a system.
 - c. Enable repositories.
 - d. Review and track entitlements.
 - e. All of the above.

► Solution

Registering Systems for Red Hat Support

Choose the correct answer to the following questions:

- ▶ **1. Which command is used to register a system without using a graphical environment?**
 - a. `rct`
 - b. `subscription-manager`
 - c. `rpm`
 - d. `yum`
- ▶ **2. Which GUI tool is used to register and subscribe a system?**
 - a. PackageKit
 - b. `gpk-application`
 - c. Red Hat Subscription Manager
 - d. `gnome-software`
- ▶ **3. Which task(s) can be performed with Red Hat Subscription Management tools?**
 - a. Register a system.
 - b. Subscribe a system.
 - c. Enable repositories.
 - d. Review and track entitlements.
 - e. All of the above.

Explaining and Investigating RPM Software Packages

Objectives

After completing this section, you should be able to explain how software is provided as RPM packages, and investigate the packages installed on the system with Yum and RPM.

Software packages and RPM

The RPM Package Manager, originally developed by Red Hat, 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 that 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 names consist of four elements (plus the `.rpm` suffix): **name-version-release.architecture**:

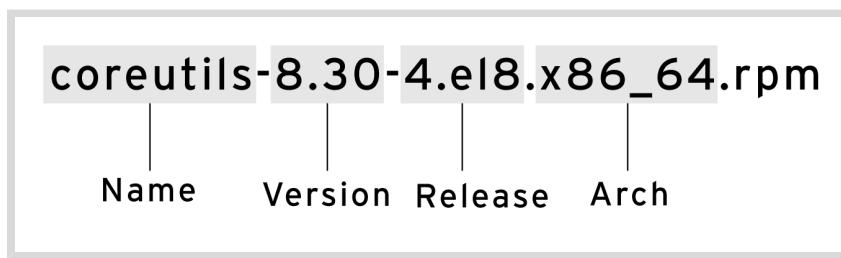


Figure 14.3: RPM file name elements

- NAME is one or more words describing the contents (coreutils).
- VERSION is the version number of the original software (8.30).
- 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 (4.el8).
- ARCH is the processor architecture the package was compiled to run on. **noarch** indicates that this package's contents are not architecture-specific (as opposed to **x86_64** for 64-bit, **aarch64** for 64-bit ARM, and so on).

Only the package name is required for installing packages from repositories. If multiple versions exist, the package with the higher version number is installed. If multiple releases of a single version exist, the package with the higher release number is 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, and arch; a summary and description of the package; whether it requires other packages to be installed; licensing; a package change log; and other details.

- Scripts that may run when this package is installed, updated, or removed, or are triggered when other packages are installed, updated, or removed.

Typically, software providers digitally sign RPM packages using GPG keys (Red Hat digitally signs all packages it releases). The RPM system verifies package integrity by confirming that the package was signed by the appropriate GPG key. The RPM system refuses to install a package if the GPG signature does not match.

Updating Software with RPM Packages

Red Hat generates a complete RPM package to update software. An administrator installing that package gets only the most recent version of the package. Red Hat does not require that older packages be installed and then patched. To update software, RPM removes the older version of the package and installs the new version. Updates usually retain configuration files, but the packager of the new version defines the exact behavior.

In most cases, only one version or release of a package may be installed at a time. However, if a package is built so that there are no conflicting file names, then multiple versions may be installed. The most important example of this is 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 and bootable.

Examining RPM Packages

The **rpm** utility is a low-level tool that can get information about the contents of package files and installed packages. By default, it gets information from the local database of installed packages. However, you can use the **-p** option to specify that you want to get information about a downloaded package file. You might want to do this in order to inspect the contents of the package file before installing it.

The general form of a query is:

- **rpm -q [select-options] [query-options]**

RPM queries: General information about installed packages

- **rpm -qa**: List all installed packages
- **rpm -qf FILENAME**: Find out what package provides FILENAME

```
[user@host ~]$ rpm -qf /etc/yum.repos.d  
redhat-release-8.0-0.39.el8.x86_64
```

RPM queries: Information about specific packages

- **rpm -q**: List what version of the package is currently installed

```
[user@host ~]$ rpm -q yum  
yum-4.0.9.2-4.el8.noarch
```

- **rpm -qi**: Get detailed information about the package
- **rpm -ql**: List the files installed by the package

```
[user@host ~]$ rpm -ql yum
/etc/yum.conf
/etc/yum/pluginconf.d
/etc/yum/protected.d
/etc/yum/vars
/usr/bin/yum
/usr/share/man/man1/yum-aliases.1.gz
/usr/share/man/man5/yum.conf.5.gz
/usr/share/man/man8/yum-shell.8.gz
/usr/share/man/man8/yum.8.gz
```

- **rpm -qc**: List just the configuration files installed by the package

```
[user@host ~]$ rpm -qc openssh-clients
/etc/ssh/ssh_config
/etc/ssh/ssh_config.d/05-redhat.conf
```

- **rpm -qd**: List just the documentation files installed by the package

```
[user@host ~]$ rpm -qd openssh-clients
/usr/share/man/man1/scp.1.gz
/usr/share/man/man1/sftp.1.gz
/usr/share/man/man1/ssh-add.1.gz
/usr/share/man/man1/ssh-agent.1.gz
/usr/share/man/man1/ssh-copy-id.1.gz
/usr/share/man/man1/ssh-keyscan.1.gz
/usr/share/man/man1/ssh.1.gz
/usr/share/man/man5/ssh_config.5.gz
/usr/share/man/man8/ssh-pkcs11-helper.8.gz
```

- **rpm -q --scripts**: List shell scripts that run before or after the package is installed or removed

```
[user@host ~]$ 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
```

- **rpm -q --changelog**: list change information for the package

```
[user@host ~]$ rpm -q --changelog audit
* Wed Jan 09 2019 Steve Grubb <sgrubb@redhat.com> 3.0-0.10.20180831git0047a6c
  resolves: rhbz#1655270] Message "audit: backlog limit exceeded" reported
  - Fix annobin failure

* Fri Dec 07 2018 Steve Grubb <sgrubb@redhat.com> 3.0-0.8.20180831git0047a6c
  resolves: rhbz#1639745 - build requires go-toolset-7 which is not available
  resolves: rhbz#1643567 - service audidd stop exits prematurely
  resolves: rhbz#1616428 - Update git snapshot of audit package
  - Remove static libs subpackage
  ...output omitted...
```

Querying local package files:

```
[user@host ~]$ ls -l wonderwidgets-1.0-4.x86_64.rpm
-rw-rw-r-- 1 user user 257 Mar 13 20:06 wonderwidgets-1.0-4.x86_64.rpm
[user@host ~]$ rpm -qlp 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
```

Installing RPM Packages

The **rpm** command can also be used to install an RPM package that you have downloaded to your local directory.

```
[root@host ~]# rpm -ivh wonderwidgets-1.0-4.x86_64.rpm
Verifying... ###### [100%]
Preparing... ###### [100%]
Updating / installing...
 1:wonderwidgets-1.0-4 ###### [100%]
[root@host ~]#
```

However, the next section of this chapter will discuss a more powerful tool for managing RPM installation and updates from the command line, **yum**.

**Warning**

Be careful when installing packages from third parties, not just because of the software that they may install, but because the RPM package may include arbitrary scripts that run as the **root** user as part of the installation process.

**Note**

You can extract files from an RPM package file without installing the package. The **rpm2cpio** utility can pass the contents of the RPM to a special archiving tool called **cpio**, which can extract all files or individual files.

Pipe the output of **rpm2cpio PACKAGEFILE.rpm** into **cpio -id**, to extract all files stored in the RPM package. Subdirectory trees are created as needed, relative to the current working directory.

```
[user@host tmp-extract]$ rpm2cpio wonderwidgets-1.0-4.x86_64.rpm | cpio -id
```

Individual files are extracted by specifying the path of the file:

```
[user@host ~]$ rpm2cpio wonderwidgets-1.0-4.x86_64.rpm | cpio -id "*txt"
11 blocks
[user@host ~]$ ls -l usr/share/doc/wonderwidgets-1.0/
total 4
-rw-r--r--. 1 user user 76 Feb 13 19:27 README.txt
```

Summary of RPM Query Commands

Installed packages can be queried directly with the **rpm** command. Add the **-q** option to query a package file before installation.

Command	Task
rpm -qa	List all RPM packages currently installed
rpm -q NAME	Display the version of NAME installed on the system
rpm -qi NAME	Display detailed information about a package
rpm -ql NAME	List all files included in a package
rpm -qc NAME	List configuration files included in a package
rpm -qd NAME	List documentation files included in a package
rpm -q --changelog NAME	Show a short summary of the reason for a new package release
rpm -q --scripts NAME	Display the shell scripts run on package installation, upgrade, or removal



References

rpm(8), **rpm2cpio(8)**, **cpio(1)**, and **rpmkeys(8)** man pages

► Guided Exercise

Explaining and Investigating RPM Software Packages

In this exercise, you will gather information about a package from a third party, extract files from it for inspection, and then install it on a server.

Outcomes

You should be able to install a package not provided by software repositories on a server.

Before You Begin

Log in as the **student** user on **workstation** using **student** as the password.

From **workstation** run the **lab software-rpm start** command. The script runs a start script that determines whether the host, **servera**, is reachable on the network. The script also downloads the *rhcsa-script-1.0.0-1.noarch.rpm* package in the **/home/student** directory on **servera**.

```
[student@workstation ~]$ lab software-rpm start
```

- 1. Use the **ssh** command to log in to **servera** as the **student** user.

```
[student@workstation ~]$ ssh student@servera  
...output omitted...  
[student@servera ~]$
```

- 2. View package information and list files in the *rhcsa-script-1.0.0-1.noarch.rpm* package. Also view the script that runs when the package is installed or uninstalled.

- 2.1. View information for the *rhcsa-script-1.0.0-1.noarch.rpm* package.

```
[student@servera ~]$ rpm -q -p rhcsa-script-1.0.0-1.noarch.rpm -i  
Name      : rhcsa-script  
Version   : 1.0.0  
Release   : 1  
Architecture: noarch  
Install Date: (not installed)  
Group     : System  
Size      : 1056  
License   : GPL  
Signature : (none)  
Source RPM : rhcsa-script-1.0.0-1.src.rpm  
Build Date : Wed 06 Mar 2019 03:59:46 PM IST  
Build Host : foundation0.ilt.example.com  
Relocations : (not relocatable)  
Packager  : Snehangshu Karmakar
```

```
URL          : http://example.com
Summary      : RHCSA Practice Script
Description  :
A RHCSA practice script.
The package changes the motd.
```

- 2.2. List files in the *rhcsa-script-1.0.0-1.noarch.rpm* package.

```
[student@servera ~]$ rpm -q -p rhcsa-script-1.0.0-1.noarch.rpm -l
/opt/rhcsa-script/mymotd
```

- 2.3. View the script that runs when the *rhcsa-script-1.0.0-1.noarch.rpm* package is installed or uninstalled.

```
[student@servera ~]$ rpm -q -p rhcsa-script-1.0.0-1.noarch.rpm --scripts
preinstall scriptlet (using /bin/sh):
if [ "$1" == "2" ]; then
    if [ -e /etc/motd.orig ]; then
        mv -f /etc/motd.orig /etc/motd
    fi
fi
postinstall scriptlet (using /bin/sh):
...output omitted...
```

- 3. Extract the contents of the *rhcsa-script-1.0.0-1.noarch.rpm* package in the **/home/student** directory.

- 3.1. Use the **rpm2cpio** and **cpio -tv** commands to list the files in the *rhcsa-script-1.0.0-1.noarch.rpm* package.

```
[student@servera ~]$ rpm2cpio rhcsa-script-1.0.0-1.noarch.rpm | cpio -tv
-rw-r--r--  1 root      root     1056 Mar  6 15:59 ./opt/rhcsa-script/mymotd
3 blocks
```

- 3.2. Extract all files from the *rhcsa-script-1.0.0-1.noarch.rpm* package in the **/home/student** directory. Use the **rpm2cpio** and **cpio -idv** commands to extract the files and create the leading directories where needed in verbose mode.

```
[student@servera ~]$ rpm2cpio rhcsa-script-1.0.0-1.noarch.rpm | cpio -idv
./opt/rhcsa-script/mymotd
3 blocks
```

- 3.3. List to verify the extracted files in the **/home/student/opt** directory.

```
[student@servera ~]$ ls -lR opt
opt:
total 0
drwxrwxr-x. 2 student student 20 Mar  7 14:44 rhcsa-script

opt/rhcsa-script:
total 4
-rw-r--r--. 1 student student 1056 Mar  7 14:44 mymota
```

- ▶ 4. Install the *rhcsa-script-1.0.0-1.noarch.rpm* package. Use the **sudo** command to gain superuser privileges to install the package.
- 4.1. Use the **sudo rpm -ivh** command to install the *rhcsa-script-1.0.0-1.noarch.rpm* RPM package.

```
[student@servera ~]$ sudo rpm -ivh rhcsa-script-1.0.0-1.noarch.rpm
[sudo] password for student: student
Verifying...                                         #####[100%]
Preparing...                                         #####[100%]
Updating / installing...
 1:rhcsa-script-1.0.0-1                           #####[100%]
[student@servera ~]$
```

- 4.2. Use the **rpm** command to verify that the package is installed.

```
[student@servera ~]$ rpm -q rhcsa-script
rhcsa-script-1.0.0-1.noarch
```

- ▶ 5. Exit from **servera**.

```
[student@servera ~]$ exit
logout
Connection to servera closed.
[student@workstation ~]$
```

Finish

On **workstation**, run the **lab software-rpm finish** script to complete this exercise. This script removes all the packages installed on **servera** during the exercise.

```
[student@workstation ~]$ lab software-rpm finish
```

This concludes the guided exercise.

Installing and Updating Software Packages with Yum

Objectives

After completing this section, you should be able to find, install, and update software packages, using the **yum** command.

Managing Software Packages with Yum

The low-level **rpm** command can be used to install packages, but it is not designed to work with package repositories or resolve dependencies from multiple sources automatically.

Yum is designed to be a better system for managing RPM-based software installation and updates. The **yum** command allows you to install, update, remove, and get information about software packages and their dependencies. You can get a history of transactions performed and work with multiple Red Hat and third-party software repositories.

Finding Software with Yum

- **yum help** displays usage information.
- **yum list** displays installed and available packages.

```
[user@host ~]$ yum list 'http*'
Available Packages
http-parser.i686          2.8.0-2.el8           rhel8-appstream
http-parser.x86_64          2.8.0-2.el8           rhel8-appstream
httpcomponents-client.noarch 4.5.5-4.module+el8+2452+b359bfcd rhel8-appstream
httpcomponents-core.noarch   4.4.10-3.module+el8+2452+b359bfcd rhel8-appstream
httpd.x86_64                2.4.37-7.module+el8+2443+605475b7 rhel8-appstream
httpd-devel.x86_64          2.4.37-7.module+el8+2443+605475b7 rhel8-appstream
httpd-filesystem.noarch     2.4.37-7.module+el8+2443+605475b7 rhel8-appstream
httpd-manual.noarch         2.4.37-7.module+el8+2443+605475b7 rhel8-appstream
httpd-tools.x86_64          2.4.37-7.module+el8+2443+605475b7 rhel8-appstream
```

- **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**:

```
[user@host ~]$ yum search all 'web server'
===== Summary & Description Matched: web server =====
pcp-pmda-weblog.x86_64 : Performance Co-Pilot (PCP) metrics from web server logs
nginx.x86_64 : A high performance web server and reverse proxy server
===== Summary Matched: web server =====
libcurl.x86_64 : A library for getting files from web servers
libcurl.i686 : A library for getting files from web servers
libcurl.x86_64 : A library for getting files from web servers
===== Description Matched: web server =====
httpd.x86_64 : Apache HTTP Server
```

```
git-instaweb.x86_64 : Repository browser in gitweb
...output omitted...
```

- **yum info PACKAGE NAME** returns detailed information about a package, including the disk space needed for installation.

To get information on the Apache HTTP Server:

```
[user@host ~]$ yum info httpd
Available Packages
Name        : httpd
Version     : 2.4.37
Release     : 7.module+el8+2443+605475b7
Arch        : x86_64
Size        : 1.4 M
Source      : httpd-2.4.37-7.module+el8+2443+605475b7.src.rpm
Repo        : rhel8-appstream
Summary     : Apache HTTP Server
URL         : https://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 path name specified (which often include wildcard characters).

To find packages that provide the **/var/www/html** directory, use:

```
[user@host ~]$ yum provides /var/www/html
httpd-filesystem-2.4.37-7.module+el8+2443+605475b7.noarch : The basic directory
layout for the Apache HTTP server
Repo        : rhel8-appstream
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.

```
[user@host ~]$ yum install httpd
Dependencies resolved.
=====
Package          Arch    Version       Repository      Size
=====
Installing:
  httpd           x86_64  2.4.37-7.module...
  apr             x86_64  1.6.3-8.el8      rhel8-appstream 125 k
  apr-util        x86_64  1.6.1-6.el8      rhel8-appstream 105 k
...output omitted...
Transaction Summary
=====
Install 9 Packages
```

```
Total download size: 2.0 M
Installed size: 5.4 M
Is this ok [y/N]: y
Downloading Packages:
(1/9): apr-util-bdb-1.6.1-6.el8.x86_64.rpm           464 kB/s | 25 kB   00:00
(2/9): apr-1.6.3-8.el8.x86_64.rpm                     1.9 MB/s | 125 kB  00:00
(3/9): apr-util-1.6.1-6.el8.x86_64.rpm               1.3 MB/s | 105 kB  00:00
...output omitted...
Total                                         8.6 MB/s | 2.0 MB  00:00

Running transaction check
Transaction check succeeded.

Running transaction test
Transaction test succeeded.

Running transaction
  Preparing          :                           1/1
  Installing         : apr-1.6.3-8.el8.x86_64      1/9
  Running scriptlet: apr-1.6.3-8.el8.x86_64      1/9
  Installing         : apr-util-bdb-1.6.1-6.el8.x86_64 2/9
...output omitted...
Installed:
  httpd-2.4.37-7.module+el8+2443+605475b7.x86_64 apr-util-bdb-1.6.1-6.el8.x86_64
  apr-util-openssl-1.6.1-6.el8.x86_64             apr-1.6.3-8.el8.x86_64
...output omitted...
Complete!
```

- **yum update PACKAGE NAME** obtains and installs a newer version of the specified 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 installs all relevant updates.

```
[user@host ~]$ sudo 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 only shows the kernel version and release, and the **-a** option shows the kernel release and additional information.

```
[user@host ~]$ yum list kernel
Installed Packages
kernel.x86_64        4.18.0-60.el8      @anaconda
kernel.x86_64        4.18.0-67.el8      @rhel-8-for-x86_64-baseos-htb-rpms
[user@host ~]$ uname -r
4.18.0-60.el8.x86_64
[user@host ~]$ uname -a
Linux host.lab.example.com 4.18.0-60.el8.x86_64 #1 SMP Fri Jan 11 19:08:11 UTC
2019 x86_64 x86_64 x86_64 GNU/Linux
```

- **yum remove PACKAGE_NAME** removes an installed software package, including any supported packages.

```
[user@host ~]$ sudo yum remove httpd
```

**Warning**

The **yum remove** command removes the packages listed and any package that requires the packages being removed (and packages which require those packages, and so on). This can lead to unexpected removal of packages, so carefully review 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 8, there are two kinds of groups. Regular groups are collections of packages. *Environment groups* are collections of regular groups. The packages or groups provided by a group may be **mandatory** (they must be installed if the group is installed), **default** (normally installed if the group is installed), or **optional** (not installed when the group is installed, unless specifically requested).

Like **yum list**, the **yum group list** command shows the names of installed and available groups.

```
[user@host ~]$ yum group list
Available Environment Groups:
  Server with GUI
  Minimal Install
  Server
...output omitted...
Available Groups:
  Container Management
  .NET Core Development
```

```
RPM Development Tools  
...output omitted...
```

Some groups are normally installed through environment groups and are hidden by default. List these hidden groups with the **yum group list hidden** command.

- **yum group info** displays information about a group. It includes a list of mandatory, default, and optional package names.

```
[user@host ~]$ yum group info "RPM Development Tools"  
Group: RPM Development Tools  
Description: These tools include core development tools such rpmbuild.  
Mandatory Packages:  
    redhat-rpm-config  
    rpm-build  
Default Packages:  
    rpmdevtools  
Optional Packages:  
    rpmlint
```

- **yum group install** installs a group that installs its mandatory and default packages and the packages they depend on.

```
[user@host ~]$ sudo yum group install "RPM Development Tools"  
...output omitted...  
Installing Groups:  
  RPM Development Tools  
  
Transaction Summary  
=====  
Install 64 Packages  
  
Total download size: 21 M  
Installed size: 62 M  
Is this ok [y/N]: y  
...output omitted...
```

Important

The behavior of Yum groups changed starting in Red Hat Enterprise Linux 7. In RHEL 7 and later, 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 are installed upon 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, or if any default or optional packages in the group are installed. Starting in RHEL 7, a group is considered to be installed *only if* **yum group install** was used to install it. The command **yum group mark install GROUPNAME** can be used to mark a group as installed, and any missing packages and their dependencies are installed upon 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 and RHEL 8 command **yum group list** did not.

Viewing transaction history

- All install and remove transactions are logged in **/var/log/dnf.rpm.log**.

```
[user@host ~]$ tail -5 /var/log/dnf.rpm.log
2019-02-26T18:27:00Z SUBDEBUG Installed: rpm-build-4.14.2-9.el8.x86_64
2019-02-26T18:27:01Z SUBDEBUG Installed: rpm-build-4.14.2-9.el8.x86_64
2019-02-26T18:27:01Z SUBDEBUG Installed: rpmdevtools-8.10-7.el8.noarch
2019-02-26T18:27:01Z SUBDEBUG Installed: rpmdevtools-8.10-7.el8.noarch
2019-02-26T18:38:40Z INFO --- logging initialized ---
```

- yum history** displays a summary of install and remove transactions.

ID	Command line	Date and time	Action(s)	Altered
7	group install RPM Devolo	2019-02-26 13:26	Install	65
6	update kernel	2019-02-26 11:41	Install	4
5	install httpd	2019-02-25 14:31	Install	9
4	-y install @base firewall	2019-02-04 11:27	Install	127 EE
3	-C -y remove firewalld -	2019-01-16 13:12	Removed	11 EE
2	-C -y remove linux-firmware	2019-01-16 13:12	Removed	1
1		2019-01-16 13:05	Install	447 EE

- The **history undo** option reverses a transaction.

```
[user@host ~]$ sudo yum history undo 5
Undoing transaction 7, from Tue 26 Feb 2019 10:40:32 AM EST
Install apr-1.6.3-8.el8.x86_64                                     @rhel8-appstream
Install apr-util-1.6.1-6.el8.x86_64                                    @rhel8-appstream
Install apr-util-bdb-1.6.1-6.el8.x86_64                                @rhel8-appstream
Install apr-util-openssl-1.6.1-6.el8.x86_64                            @rhel8-appstream
Install httpd-2.4.37-7.module+el8+2443+605475b7.x86_64                  @rhel8-appstream
...output omitted...
```

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 group list
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 group install GROUP NAME
Update all packages	yum update
Remove a package	yum remove PACKAGE NAME
Display transaction history	yum history



References

yum(1) and **yum.conf(5)** man pages

For more information, refer to the *Managing software packages* chapter in the *Red Hat Enterprise Linux 8 Configuring basic system settings Guide* at
https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/8/html-single/configuring_basic_system_settings/index#managing-software-packages_configuring-basic-system-settings

► Guided Exercise

Installing and Updating Software Packages with Yum

In this exercise, you will install and remove packages and package groups.

Outcomes

You should be able to install and remove packages with dependencies.

Before You Begin

Log in as the **student** user on **workstation** using **student** as the password.

From **workstation** run the **lab software-yum start** command. The command runs a start script that determines if the host, **servera**, is reachable on the network.

```
[student@workstation ~]$ lab software-yum start
```

- ▶ 1. Use the **ssh** command to log in to **servera** as the **student** user. The systems are configured to use SSH keys for authentication, so a password is not required to log in to **servera**.

```
[student@workstation ~]$ ssh student@servera  
...output omitted...  
[student@servera ~]$
```

- ▶ 2. Use the **sudo -i** command to switch to the **root** user at the shell prompt.

```
[student@servera ~]$ sudo -i  
Password: student  
[root@servera ~]#
```

- ▶ 3. Search for a specific package.

- 3.1. Attempt to run the command **guile**. You should find that it is not installed.

```
[root@servera ~]# guile  
-bash: guile: command not found
```

- 3.2. Use the **yum search** command to search for packages that have **guile** as part of their name or summary.

```
[root@servera ~]# yum search guile
=====
Name Exactly Matched: guile =====
guile.i686 : A GNU implementation of Scheme for application extensibility
guile.x86_64 : A GNU implementation of Scheme for application extensibility
```

- 3.3. Use the **yum info** command to find out more information about the **guile** package.

```
[root@servera ~]# yum info guile
Available Packages
Name        : guile
Epoch       : 5
Version     : 2.0.14
Release     : 7.el8
...output omitted...
```

- 4. Use the **yum install** command to install the **guile** package.

```
[root@servera ~]# yum install guile
...output omitted...
Dependencies resolved.
=====
Package      Arch    Version      Repository      Size
=====
Installing:
guile        x86_64  5:2.0.14-7.el8   rhel-8.2-for-x86_64-appstream-rpms 3.5 M
Installing dependencies:
gc            x86_64  7.6.4-3.el8      rhel-8.2-for-x86_64-appstream-rpms 109 k
libatomic_ops x86_64  7.6.2-3.el8      rhel-8.2-for-x86_64-appstream-rpms 38 k
libtool-ltdl x86_64  2.4.6-25.el8     rhel-8.2-for-x86_64-baseos-rpms   58 k

Transaction Summary
=====
Install 4 Packages

Total download size: 3.7 M
Installed size: 12 M
Is this ok [y/N]: y
...output omitted...
Complete!
```

- 5. Remove packages.

- 5.1. Use the **yum remove** command to remove the **guile** package, but respond with **no** when prompted. How many packages would be removed?

```
[root@servera ~]# yum remove guile
...output omitted...
Dependencies resolved.
=====
Package      Arch    Version      Repository      Size
=====
```

Chapter 14 | Installing and Updating Software Packages

```
Removing:
guile           x86_64 5:2.0.14-7.el8  @rhel-8.2-for-x86_64-appstream-rpms 12 M
Removing unused dependencies:
gc               x86_64 7.6.4-3.el8      @rhel-8.2-for-x86_64-appstream-rpms 221 k
libatomic_ops    x86_64 7.6.2-3.el8      @rhel-8.2-for-x86_64-appstream-rpms 75 k
libtool-ltdl    x86_64 2.4.6-25.el8     @rhel-8.2-for-x86_64-baseos-rpms   69 k

Transaction Summary
=====
Remove 4 Packages

Freed space: 12 M
Is this ok [y/N]: n
Operation aborted.
```

- 5.2. Use the **yum remove** command to remove the **gc** package, but respond with **no** when prompted. How many packages would be removed?

```
[root@servera ~]# yum remove gc
...output omitted...
Dependencies resolved.
=====
Package      Arch Version       Repository      Size
=====
Removing:
gc           x86_64 7.6.4-3.el8  @rhel-8.2-for-x86_64-appstream-rpms 221 k
Removing dependent packages:
guile        x86_64 5:2.0.14-7.el8  @rhel-8.2-for-x86_64-appstream-rpms 12 M
Removing unused dependencies:
libatomic_ops x86_64 7.6.2-3.el8      @rhel-8.2-for-x86_64-appstream-rpms 75 k
libtool-ltdl x86_64 2.4.6-25.el8     @rhel-8.2-for-x86_64-baseos-rpms   69 k

Transaction Summary
=====
Remove 4 Packages

Freed space: 12 M
Is this ok [y/N]: n
Operation aborted.
```

- 6. Gather information about the “Security Tools” component group and install it on **servera**.

- 6.1. Use the **yum group list** command to list all available component groups.

```
[root@servera ~]# yum group list
```

- 6.2. Use the **yum group info** command to find out more information about the **Security Tools** component group, including a list of included packages.

```
[root@servera ~]# yum group info "Security Tools"
Group: Security Tools
Description: Security tools for integrity and trust verification.
Default Packages:
```

```

scap-security-guide
Optional Packages:
  aide
  hmaccalc
  openscap
  openscap-engine-sce
  openscap-utils
  scap-security-guide-doc
  scap-workbench
  tpm-quote-tools
  tpm-tools
  tpm2-tools
  trousers
  udica

```

- 6.3. Use the **yum group install** command to install the **Security Tools** component group.

```

[root@servera ~]# yum group install "Security Tools"
Dependencies resolved.
=====
Package           Arch    Version      Repository      Size
=====
Installing group/module packages:
scap-security-guide noarch 0.1.48-7.el8 rhel-8-for-x86_64-appstream-rpms 6.9 M
Installing dependencies:
GConf2            x86_64  3.2.6-22.el8 rhel-8-for-x86_64-appstream-rpms 1.0 M
...output omitted...

Transaction Summary
=====
Install 6 Packages

Total download size: 12 M
Installed size: 247 M
Is this ok [y/N]: y
...output omitted...
Installed:
  GConf2-3.2.6-22.el8.x86_64          libxslt-1.1.32-4.el8.x86_64
  openscap-1.3.2-6.el8.x86_64         openscap-scanner-1.3.2-6.el8.x86_64
  scap-security-guide-0.1.48-7.el8.noarch  xml-common-0.6.3-50.el8.noarch

Complete!

```

► 7. Explore the history and undo options of **yum**.

- 7.1. Use the **yum history** command to display recent **yum** history.

```

[root@servera ~]# yum history
ID      | Command line          | Date and time      | Action(s)      | Altered
-----
6 | group install Security T | 2019-02-26 17:11 | Install       |    7
5 | install guile          | 2019-05-26 17:05 | Install       |    4
4 | -y install @base firewall | 2019-02-04 11:27 | Install       | 127 EE

```

```

3 | -C -y remove firewalld - | 2019-01-16 13:12 | Removed      | 11 EE
2 | -C -y remove linux-firmw | 2019-01-16 13:12 | Removed      | 1
1 |                           | 2019-01-16 13:05 | Install       | 447 EE

```

On your system, the history is probably different.

- 7.2. Use the **yum history info** command to confirm that the last transaction is the group installation. In the following command, replace the transaction ID by the one from the preceding step.

```
[root@servera ~]# yum history info 6
Transaction ID : 6
Begin time      : Tue 26 Feb 2019 05:11:25 PM EST
Begin rpmsdb    : 563:bf48c46156982a78e290795400482694072f5ebb
End time        : Tue 26 Feb 2019 05:11:33 PM EST (8 seconds)
End rpmsdb      : 623:bf25b424ccf451dd0a6e674fb48e497e66636203
User            : Student User <student>
Return-Code     : Success
Releasever     : 8
Command Line   : group install Security Tools
Packages Altered:
  Install libxslt-1.1.32-4.el8.x86_64          @rhel-8.2-for-x86_64-baseos-rpms
  Install xml-common-0.6.3-50.el8.noarch        @rhel-8.2-for-x86_64-baseos-rpms
...output omitted...
```

- 7.3. Use the **yum history undo** command to remove the set of packages that were installed when the **guile** package was installed. On your system, find the correct transaction ID from the output of the **yum history** command, and then use that ID in the following command.

```
[root@servera ~]# yum history undo 5
```

- 8. Log out of the **servera** system.

```
[root@servera ~]# exit
logout
[student@servera ~]$ exit
Connection to servera closed.
[student@workstation ~]$
```

Finish

On **workstation**, run the **lab software-yum finish** script to finish this exercise.

```
[student@workstation ~]$ lab software-yum finish
```

This concludes the guided exercise.

Enabling Yum Software Repositories

Objectives

After completing this section, you should be able to enable and disable use of Red Hat or third-party Yum repositories by a server.

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:

```
[user@host ~]$ yum repolist all
...output omitted...
rhel-8-for-x86_64-appstream-debug-rpms    ... AppStream (Debug RPMs)  disabled
rhel-8-for-x86_64-appstream-rpms           ... AppStream (RPMs)        enabled:
5,045
rhel-8-for-x86_64-appstream-source-rpms   ... AppStream (Source RPMs) disabled
rhel-8-for-x86_64-baseos-debug-rpms       ... BaseOS (Debug RPMs)   enabled:
2,270
rhel-8-for-x86_64-baseos-rpms            ... BaseOS (RPMs)         enabled:
1,963
...output omitted...
```

The **yum config-manager** command can be used to enable or disable repositories. To enable a repository, the command sets the **enabled** parameter to **1**. For example, the following command enables the **rhel-8-server-debug-rpms** repository:

```
[user@host ~]$ yum config-manager --enable rhel-8-server-debug-rpms
Updating Subscription Management repositories.
=====
repo: rhel-8-for-x86_64-baseos-debug-rpms =====
[rhel-8-for-x86_64-baseos-debug-rpms]
bandwidth = 0
baseurl = [https://cdn.redhat.com/content/dist/rhel8/8/x86_64/baseos/debug]
cachedir = /var/cache/dnf
cost = 1000
deltarpm = 1
deltarpm_percentage = 75
enabled = 1
...output omitted...
```

Non-Red Hat sources provide software through third-party repositories, which can be accessed by the **yum** command from a website, FTP server, or the local file system. For example, Adobe provides some of its software for Linux through a Yum repository. In a Red Hat classroom, the `content.example.com` classroom server hosts Yum repositories.

To enable support for a new third-party repository, create a file in the `/etc/yum.repos.d/` directory. Repository configuration files must end with a `.repo` extension. 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.

Creating Yum Repositories

Create Yum repositories with the **yum config-manager** command.

The following command creates a file named **/etc/yum.repos.d/dl.fedoraproject.org_pub_epel_8_Everything_x86_64.repo** with the output shown.

```
[user@host ~]$ yum config-manager \
--add-repo="https://dl.fedoraproject.org/pub/epel/8/Everything/x86_64/"
Adding repo from: https://dl.fedoraproject.org/pub/epel/8/Everything/x86_64/

[dl.fedoraproject.org_pub_epel_8_Everything_x86_64_]
name=created by yum config-manager from https://dl.fedoraproject.org/pub/epel/8/
Everything/x86_64/
baseurl=https://dl.fedoraproject.org/pub/epel/8/Everything/x86_64/
enabled=1
```

Modify this file to provide customized values and location of a GPG key. Keys are stored in various locations on the remote repository site, such as, <http://dl.fedoraproject.org/pub/epel/RPM-GPG-KEY-EPEL-8>. Administrators should download the key to a local file rather than allowing **yum** to retrieve the key from an external source. For example:

```
[EPEL]
name=EPEL 8
baseurl=https://dl.fedoraproject.org/pub/epel/8/Everything/x86_64/
enabled=1
gpgcheck=1
gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-EPEL-8
```

RPM Configuration Packages for Local Repositories

Some repositories provide a configuration file and GPG public key as part of an RPM package that can be downloaded and installed using the **yum localinstall** command. For example, the volunteer project called Extra Packages for Enterprise Linux (EPEL) provides software not supported by Red Hat but compatible with Red Hat Enterprise Linux.

The following command installs the Red Hat Enterprise Linux 8 EPEL repository package:

```
[user@host ~]$ rpm --import \
http://dl.fedoraproject.org/pub/epel/RPM-GPG-KEY-EPEL-8
[user@host ~]$ yum install \
https://dl.fedoraproject.org/pub/epel/epel-release-latest-8.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.

```
[user@host ~]$ cat /etc/yum.repos.d/epel.repo
[epel]
name=Extra Packages for Enterprise Linux $releasever - $basearch
#baseurl=https://download.fedoraproject.org/pub/epel/$releasever/Everything/
$basearch
```

```
metalink=https://mirrors.fedoraproject.org/metalink?repo=epel-$releasever&arch=$basearch&infra=$infra&content=$contentdir  
enabled=1  
gpgcheck=1  
gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-EPEL-8  
  
[epel-debuginfo]  
name=Extra Packages for Enterprise Linux $releasever - $basearch - Debug  
#baseurl=https://download.fedoraproject.org/pub/epel/$releasever/Everything/  
$basearch/debug  
metalink=https://mirrors.fedoraproject.org/metalink?repo=epel-debug-  
$releasever&arch=$basearch&infra=$infra&content=$contentdir  
enabled=0  
gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-EPEL-8  
gpgcheck=1  
  
[epel-source]  
name=Extra Packages for Enterprise Linux $releasever - $basearch - Source  
#baseurl=https://download.fedoraproject.org/pub/epel/$releasever/Everything/SRPMS  
metalink=https://mirrors.fedoraproject.org/metalink?repo=epel-source-  
$releasever&arch=$basearch&infra=$infra&content=$contentdir  
enabled=0  
gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-EPEL-8  
gpgcheck=1
```

To define a repository, but not search it by default, insert the **enabled=0** parameter. Repositories can be enabled and disabled persistently with **yum config-manager** command or temporarily with **yum** command options, **--enablerepo= PATTERN** and **--disablerepo= PATTERN**.



Warning

Install the RPM GPG key before installing signed packages. This verifies that the packages belong to a key which has been imported. Otherwise, the **yum** command fails due to a 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

yum(1), **yum.conf(5)**, and **yum-config-manager(1)** man pages

For more information, refer to the *Managing software repositories* chapter in the *Red Hat Enterprise Linux 8 Configuring basic system settings Guide* at https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/8/html-single/configuring_basic_system_settings/index#managing-software-repositories_managing-software-packages

► Guided Exercise

Enabling Yum Software Repositories

In this exercise, you will configure your server to get packages from a remote Yum repository, then update or install a package from that repository.

Outcomes

You should be able to configure a system to obtain software updates from a classroom server and update the system to use latest packages.

Before You Begin

Log in as the **student** user on **workstation** using **student** as the password.

From **workstation** run the **lab software-repo start** command. The command runs a start script that determines whether the host **servera** is reachable on the network. The script also ensures that the **yum** package is installed.

```
[student@workstation ~]$ lab software-repo start
```

- 1. Use the **ssh** command to log in to **servera** as the **student** user.

```
[student@workstation ~]$ ssh student@servera  
...output omitted...  
[student@servera ~]$
```

- 2. Use the **sudo -i** command to switch to **root** at the shell prompt.

```
[student@servera ~]$ sudo -i  
[sudo] password for student:  
[root@servera ~]#
```

- 3. Configure software repositories on **servera** to obtain custom packages and updates from the following URL:

- Custom packages provided at http://content.example.com/rhel8.2/x86_64/rhcsa-practice/rht
- Updates of the custom packages provided at http://content.example.com/rhel8.2/x86_64/rhcsa-practice/errata

- 3.1. Use **yum config-manager** to add the custom packages repository.

```
[root@servera ~]# yum config-manager \  
--add-repo "http://content.example.com/rhel8.2/x86_64/rhcsa-practice/rht"  
Adding repo from: http://content.example.com/rhel8.2/x86_64/rhcsa-practice/rht
```

- 3.2. Examine the software repository file created by the previous command in the `/etc/yum.repos.d` directory. Use the `vim` command to edit the file and add the `gpgcheck=0` parameter to disable the GPG key check for the repository.

```
[root@servera ~]# vim \
/etc/yum.repos.d/content.example.com_rhel8.2_x86_64_rhcsa-practice_rht.repo
[content.example.com_rhel8.2_x86_64_rhcsa-practice_rht]
name=created by dnf config-manager from http://content.example.com/rhel8.2/x86_64/
rhcsa-practice/rht
baseurl=http://content.example.com/rhel8.2/x86_64/rhcsa-practice/rht
enabled=1
gpgcheck=0
```

- 3.3. Create the `/etc/yum.repos.d/errata.repo` file to enable the updates repository with the following content:

```
[rht-updates]
name=rht updates
baseurl=http://content.example.com/rhel8.2/x86_64/rhcsa-practice/errata
enabled=1
gpgcheck=0
```

- 3.4. Use the `yum repolist all` command to list all repositories on the system:

```
[root@servera ~]# yum repolist all
repo id                                repo name      status
content.example.com_rhel8.2_x86_64_rhcsa-practice_rht  created by .... enabled
rht-updates                               rht updates    enabled
...output omitted...
```

- ▶ 4. Disable the `rht-updates` software repository and install the `rht-system` package.

- 4.1. Use `yum config-manager --disable` to disable the `rht-updates` repository.

```
[root@servera ~]# yum config-manager --disable rht-updates
```

- 4.2. List, and then install, the `rht-system` package:

```
[root@servera ~]# yum list rht-system
Available Packages
rht-system.noarch 1.0.0-1 content.example.com_rhel8.2_x86_64_rhcsa-practice_rht
[root@servera ~]# yum install rht-system
Dependencies resolved.
=====
Package          Arch      Version       Repository      Size
=====
Installing:
rht-system      noarch   1.0.0-1      content..._rht  3.7 k
...output omitted...
Is this ok [y/N]: y
...output omitted...
```

```
Installed:  
rht-system-1.0.0-1.noarch  
Complete!
```

- 4.3. Verify that the *rht-system* package is installed, and note the version number of the package.

```
[root@servera ~]# yum list rht-system  
Installed Packages  
rht-system.noarch 1.0.0-1 @content.example.com_rhel8.2_x86_64_rhcsa-practice_rht
```

- 5. Enable the **rht-updates** software repository and update all relevant software packages.

- 5.1. Use **yum config-manager --enable** to enable the **rht-updates** repository.

```
[root@servera ~]# yum config-manager --enable rht-updates
```

- 5.2. Use the **yum update** command to update all software packages on **servera**.

```
[root@servera ~]# yum update  
Dependencies resolved.  
=====  
Package          Arch      Version       Repository      Size  
=====  
Upgrading:  
  rht-system      x86_64    1.0.0-2.el7   rht-updates   3.9 k  
...output omitted...  
Is this ok [y/N]: y  
...output omitted...  
Complete!
```

- 5.3. Verify that the *rht-system* package is upgraded, and note the version number of the package.

```
[root@servera ~]# yum list rht-system  
Installed Packages  
rht-system.noarch 1.0.0-2.el7      @rht-updates
```

- 6. Exit from **servera**.

```
[root@servera ~]# exit  
logout  
[student@servera ~]$ exit  
logout  
Connection to servera closed.  
[student@workstation ~]$
```

Finish

On **workstation**, run the **lab software-repo finish** script to finish this exercise. This script removes all the software repositories and packages installed on **servera** during the exercise.

```
[student@workstation ~]$ lab software-repo finish
```

This concludes the guided exercise.

Managing Package Module Streams

Objectives

After completing this section, you should be able to:

- Explain how modules allow installation of specific versions of software.
- How to list, enable, and switch module streams.
- Install, and update packages from a module.

Introduction to Application Stream

Red Hat Enterprise Linux 8.0 introduces the concept of Application Streams. Multiple versions of user space components shipped with the distribution are now delivered at the same time. They may be updated more frequently than the core operating system packages. This provides you with greater flexibility to customize Red Hat Enterprise Linux without impacting the underlying stability of the platform or specific deployments.

Traditionally, managing alternate versions of an application's software package and its related packages meant maintaining different repositories for each different version. For developers who wanted the latest version of an application and administrators who wanted the most stable version of the application, this created a situation that was tedious to manage. This process is simplified in Red Hat Enterprise Linux 8 using a new technology called *Modularity*. Modularity allows a single repository to host multiple versions of an application's package and its dependencies.

Red Hat Enterprise Linux 8 content is distributed through two main software repositories: *BaseOS* and *Application Stream (AppStream)*.

BaseOS

The BaseOS repository provides the core operating system content for Red Hat Enterprise Linux as RPM packages. BaseOS components have a life cycle identical to that of content in previous Red Hat Enterprise Linux releases.

Application Stream

The Application Stream repository provides content with varying life cycles as both modules and traditional packages. Application Stream contains necessary parts of the system, as well as a wide range of applications previously available as a part of Red Hat Software Collections and other products and programs.



Important

Both BaseOS and AppStream are a necessary part of a Red Hat Enterprise Linux 8 system.

The Application Stream repository contains two types of content: *Modules* and traditional RPM packages. A module describes a set of RPM packages that belong together. Modules can contain

several streams to make multiple versions of applications available for installation. Enabling a module stream gives the system access to the RPM packages within that module stream.

Modules

A module is a set of RPM packages that are a consistent set that belong together. Typically, this is organized around a specific version of a software application or programming language. A typical module can contain packages with an application, packages with the application's specific dependency libraries, packages with documentation for the application, and packages with helper utilities.

Module Streams

Each module can have one or more module streams, which hold different versions of the content. Each of the streams receives updates independently. Think of the module stream as a virtual repository in the Application Stream physical repository.

For each module, only one of its streams can be enabled and provide its packages.

Module Profiles

Each module can have one or more profiles. A profile is a list of certain packages to be installed together for a particular use-case such as for a server, client, development, minimal install, or other.

Installing a particular module profile simply installs a particular set of packages from the module stream. You can subsequently install or uninstall packages normally. If you do not specify a profile, the module will install its *default profile*.

Managing modules using Yum

Yum version 4, new in Red Hat Enterprise Linux 8, adds support for the new modular features of Application Stream.

For handling the modular content, the **yum module** command has been added. Otherwise, **yum** works with modules much like does with regular packages.

Listing Modules

To display a list of available modules, use **yum module list**:

```
[user@host ~]$ yum module list
Red Hat Enterprise Linux 8 for x86_64 - AppStream (RPMS)
Name           Stream     Profiles   Summary
389-ds         1.4        default    389 Directory Server (base)
ant            1.10 [d]    common    [d] Java build tool
container-tools 1.0 [d]    common    [d] Common tools and dependencies for
                  container runtimes
...output omitted...
Hint: [d]efault, [e]nabled, [x]disabled, [i]nstalled
```

**Note**

Use the *Hint* at the end of the output to help determine which streams and profiles are enabled, disabled, installed, as well as which ones are the defaults.

To list the module streams for a specific module and retrieve their status:

```
[user@host ~]$ yum module list perl
Red Hat Enterprise Linux 8 for x86_64 - AppStream (RPMs)
Name Stream Profiles Summary
perl 5.24 common [d], minimal Practical Extraction and Report Language
perl 5.26 [d] common [d], minimal Practical Extraction and Report Language
```

To display details of a module:

```
[user@host ~]$ yum module info perl
Name : perl
Stream : 5.24
Version : 820190207164249
Context : ee766497
Profiles : common [d], minimal
Default profiles : common
Repo : rhel-8-for-x86_64-appstream-rpms
Summary : Practical Extraction and Report Language
...output omitted...
Artifacts : perl-4:5.24.4-403.module+el8+2770+c759b41a.x86_64
            : perl-Algorithm-Diff-0:1.1903-9.module+el8+2464+d274aed1.noarch
            : perl-Archive-Tar-0:2.30-1.module+el8+2464+d274aed1.noarch
...output omitted...
```

**Note**

Without specifying a module stream, **yum module info** shows list of packages installed by default profile of a module using the default stream. Use the *module-name:stream* format to view a specific module stream. Add the **--profile** option to display information about packages installed by each of the module's profiles. For example:

```
[user@host ~]$ yum module info --profile perl:5.24
```

Enabling Module Streams and Installing Modules

Module streams must be enabled in order to install their module. To simplify this process, when a module is installed it enables its module stream if necessary. Module streams can be enabled manually using **yum module enable** and providing the name of the module stream.

**Important**

Only one module stream may be enabled for a given module. Enabling an additional module stream will disable the original module stream.

Install a module using the default stream and profiles:

```
[user@host ~]$ sudo yum module install perl
Dependencies resolved.
=====
Package           Arch    Version      Repository          Size
=====
Installing group/module packages:
perl             x86_64  4:5.26.3-416.el8
                           rhel-8-for-x86_64-appstream-htb-rpms 72 k
Installing dependencies:
...output omitted...
Running transaction
Preparing          :                                     1/1
Installing        : perl-Exporter-5.72-396.el8.noarch      1/155
Installing        : perl-Carp-1.42-396.el8.noarch          2/155
...output omitted...
Installed:
perl-4:5.26.3-416.el8.x86_64
perl-Encode-Locale-1.05-9.el8.noarch
...output omitted...
Complete!
```

**Note**

The same results could have been accomplished by running **yum install @perl**. The @ notation informs **yum** that the argument is a module name instead of a package name.

To verify the status of the module stream and the installed profile:

```
[user@host ~]$ yum module list perl
Red Hat Enterprise Linux 8 for x86_64 - AppStream (RPMS)
Name   Stream     Profiles          Summary
perl   5.24       common, minimal  Practical Extraction and Report Language
perl   5.26 [d][e]  common [i], minimal  Practical Extraction and Report Language

Hint: [d]efault, [e]nabled, [x]disabled, [i]nstalled
```

Removing Modules and Disabling Module Streams

Removing a module removes all of the packages installed by profiles of the currently enabled module stream, and any further packages and modules that depend on these. Packages installed from this module stream not listed in any of its profiles remain installed on the system and can be removed manually.

**Warning**

Removing modules and switching module streams can be a bit tricky. Switching the stream enabled for a module is equivalent to resetting the current stream and enabling the new stream. It does not automatically change any installed packages. You have to do that manually.

Directly installing a module stream that is different than the one that is currently installed is not recommended, because upgrade scripts might run during the installation that would break things with the original module stream. That could lead to data loss or other configuration issues.

Proceed with caution.

To remove an installed module:

```
[user@host ~]$ sudo yum module remove perl
Dependencies resolved.

=====
Package           ArchVersion      Repository
Size
=====

Removing:
perl              x86_64:5.26.3-416.el8    @rhel-8.0-for-x86_64-
appstream-rpms 0

Removing unused dependencies:
...output omitted...
Running transaction
Preparing          :                               1/1
Erasing            : perl-4:5.26.3-416.el8.x86_64          1/155
Erasing            : perl-CPAN-2.18-397.el8.noarch          2/155
...output omitted...
Removed:
perl-4:5.26.3-416.el8.x86_64
dwz-0.12-9.el8.x86_64
...output omitted...
Complete!
```

After the module is removed, the module stream is still enabled. To verify the module stream is still enabled:

```
[user@host ~]$ yum module list perl
Red Hat Enterprise Linux 8 for x86_64 - AppStream (RPMs)
Name Stream      Profiles       Summary
perl 5.24        common [d], minimal  Practical Extraction and Report Language
perl 5.26 [d][e]  common [d], minimal  Practical Extraction and Report Language

Hint: [d]efault, [e]nabled, [x]disabled, [i]nstalled
```

To disable the module stream:

```
[user@host ~]$ sudo yum module disable perl
...output omitted...
Dependencies resolved.
=====
Package           Arch      Version       Repository      Size
=====
Disabling module streams:
perl              5.26
Is this ok [y/N]: y
Complete!
```

Switching Module Streams

Switching module streams generally requires upgrading or downgrading the content to a different version.

To ensure a clean switch, you should remove the modules provided by the module stream first. That will remove all the packages installed by the profiles of the module, and any modules and packages that those packages have dependencies on.

To list the packages installed from the module, in the example below the *postgresql:9.6* module is installed:

```
[user@host ~]$ sudo yum module info postgresql | grep module+el8 | \
sed 's/.*/ //g;s/\n/ /g' | xargs yum list installed
Installed Packages
postgresql.x86_64          9.6.10-1.module+el8+2470+d1bafa0e @rhel-8.0-for-
x86_64-appstream-rpms
postgresql-server.x86_64     9.6.10-1.module+el8+2470+d1bafa0e @rhel-8.0-for-
x86_64-appstream-rpms
```

Remove the packages listed from the previous command. Mark the module profiles to be uninstalled.

```
[user@host ~]$ sudo yum module remove postgresql
...output omitted...
Is this ok [y/N]: y
...output omitted...
Removed:
postgresql-server-9.6.10-1.module+el8+2470+d1bafa0e.x86_64
libpq-10.5-1.el8.x86_64  postgresql-9.6.10-1.module+el8+2470+d1bafa0e.x86_64
Complete
```

After removing the module profiles reset the module stream. Use the **yum module reset** command to reset the module stream.

```
[user@host ~]$ sudo yum module reset postgresql
=====
Package           Arch      Version       Repository      Size
=====
Resetting module streams:
postgresql        9.6
```

Transaction Summary

```
=====  
Is this ok [y/N]: y  
Complete!
```

To enable a different module stream and install the module:

```
[user@host ~]$ sudo yum module install postgresql:10
```

The new module stream will be enabled and the current stream disabled. It may be necessary to update or downgrade packages from the previous module stream that are not listed in the new profile. Use the **yum distro-sync** to perform this task if required. There may also be packages that remain installed from the previous module stream. Remove those using **yum remove**.



References

For more information, refer to the *Using AppStream* chapter in the *Red Hat Enterprise Linux 8 Installing, managing, and removing user space components Guide* at

https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/8/html-single/installing_managing_and_removing_user-space_components/index#using-appstream_using-appstream

Modularity

<https://docs.fedoraproject.org/en-US/modularity/>

► Guided Exercise

Managing Package Module Streams

In this exercise, you will list the available modules, enable a specific module stream, and install packages from that stream.

Outcomes

You should be able to:

- List installed modules and examine information of a module.
- Enable and install a module from a stream.
- Switch to a specific module stream.
- Remove and disable a module.

Before You Begin

Log in as the **student** user on **workstation** using **student** as the password.

From **workstation** run the **lab software-module start** command. The command runs a start script that determines whether the host, **servera**, is reachable on the network. The script also ensures that the required software repositories are available and installs the **postgresql:9.6** module.

```
[student@workstation ~]$ lab software-module start
```

- 1. Use the **ssh** command to log in to **servera** as the **student** user.

```
[student@workstation ~]$ ssh student@servera
...output omitted...
[student@servera ~]$
```

- 2. Switch to **root** at the shell prompt.

```
[student@servera ~]$ sudo -i
[sudo] password for student: student
[root@servera ~]#
```

- 3. List available modules, streams, and installed modules. Examine the information for the **python36** module.

- 3.1. Use the **yum module list** command to list available modules and streams.

Name	Stream	Profiles	Summary

```
...output omitted...
python27      2.7 [d]      common [d]      Python ... version 2.7
python36      3.6 [d][e]    build, common [d]  Python ... version 3.6
python38      3.8 [d]      build, common [d]  Python ... version 3.8
...output omitted...
Hint: [d]efault, [e]nabled, [x]disabled, [i]nstalled
```

- 3.2. Use the **yum module list --installed** command to list installed modules and streams.

```
[root@servera ~]# yum module list --installed
Red Hat Enterprise Linux 8.2 AppStream (dvd)
Name           Stream     Profiles          Summary
postgresql     9.6 [e]    client, server [d] [i] PostgreSQL server and client ...

Hint: [d]efault, [e]nabled, [x]disabled, [i]nstalled
```

- 3.3. Use the **yum module info** command to examine details of the *python36* module.

```
[root@servera ~]# yum module info python36
Name          : python36
Stream        : 3.6 [d][e][a]
Version       : 8010020190724083915
Context       : a920e634
Architecture   : x86_64
Profiles      : build, common [d]
Default profiles: common
Repo          : rhel-8.2-for-x86_64-appstream-rpms
Summary       : Python programming language, version 3.6
...output omitted...
Hint: [d]efault, [e]nabled, [x]disabled, [i]nstalled, [a]ctive]
```

- ▶ 4. Install the *python36* module from the **3.6** stream and the **common** profile. Verify the current status of the module.

- 4.1. Use the **yum module install** command to install the *python36* module. Use the **name:stream/profile** syntax to install the *python36* module from the **3.6** stream and the **common** profile.



Note

You can omit **/profile** to use the default profile and **:stream** to use the default stream.

```
[root@servera ~]# yum module install python36:3.6/common
...output omitted...
Is this ok [y/N]: y
...output omitted...
Complete!
```

- 4.2. Examine the current status of the *python36* module.

```
[root@servera ~]# yum module list python36
Red Hat Enterprise Linux 8.2 AppStream (dvd)
Name      Stream      Profiles          Summary
python36   3.6 [d][e]  build, common [d] [i]  Python ... version 3.6

Hint: [d]efault, [e]nabled, [x]disabled, [i]nstalled
```

- 5. Switch the *postgresql* module of the **server** profile to use the **10** stream.

- 5.1. Use the **yum module list** command to list the *postgresql* module and the stream. Notice that the *postgresql:9.6* module stream is currently installed.

```
[root@servera ~]# yum module list postgresql
Red Hat Enterprise Linux 8.2 AppStream (dvd)
Name      Stream      Profiles          Summary
postgresql 9.6 [e]  client, server [d] [i] PostgreSQL server and client ...
postgresql 10 [d]    client, server [d]    PostgreSQL server and client ...
postgresql 12        client, server [d]    PostgreSQL server and client ...

Hint: [d]efault, [e]nabled, [x]disabled, [i]nstalled
```

- 5.2. Remove and disable the *postgresql* module stream along with all the packages installed by the profile.

```
[root@servera ~]# yum module remove postgresql
...output omitted...
Is this ok [y/N]: y
...output omitted...
Removed:
  postgresql-server-9.6.10-1.module+el8+2470+d1bafa0e.x86_64
  libpq-10.5-1.el8.x86_64  postgresql-9.6.10-1.module+el8+2470+d1bafa0e.x86_64
Complete
```

- 5.3. Reset the *postgresql* module and its streams.

```
[root@servera ~]# yum module reset postgresql
=====
Package      Arch      Version      Repository      Size
=====
Resetting modules:
postgresql

Transaction Summary
=====

Is this ok [y/N]: y
Complete!
```

- 5.4. Use the **yum module install** command to switch to the *postgresql:10* module stream.

```
[root@servera ~]# yum module install postgresql:10
...output omitted...
Is this ok [y/N]: y
...output omitted...
Complete!
```

5.5. Verify that the *postgresql* module is switched to the **10** stream.

```
[root@servera ~]# yum module list postgresql
Red Hat Enterprise Linux 8.2 AppStream (dvd)
Name      Stream   Profiles          Summary
postgresql 9.6       client, server [d] PostgreSQL server and client ...
postgresql 10 [d][e] client, server [d] [i] PostgreSQL server and client ...
postgresql 12       client, server [d] PostgreSQL server and client ...

Hint: [d]efault, [e]nabled, [x]disabled, [i]nstalled
```

► 6. Remove and disable the *postgresql* module stream along with all the packages installed by the profile.

6.1. Use the **yum module remove** command to remove the *postgresql* module. The command also removes all packages installed from this module.

```
[root@servera ~]# yum module remove postgresql
...output omitted...
Is this ok [y/N]: y
...output omitted...
Complete!
```

6.2. Disable the *postgresql* module stream.

```
[root@servera ~]# yum module disable postgresql
...output omitted...
Is this ok [y/N]: y
...output omitted...
Complete!
```

6.3. Verify that the *postgresql* module stream is removed and disabled.

```
[root@servera ~]# yum module list postgresql
Red Hat Enterprise Linux 8.2 AppStream (dvd)
Name      Stream   Profiles          Summary
postgresql 9.6 [x]     client, server [d] PostgreSQL server and client ...
postgresql 10 [d][x] client, server [d] PostgreSQL server and client ...
postgresql 12 [x]     client, server [d] PostgreSQL server and client ...

Hint: [d]efault, [e]nabled, [x]disabled, [i]nstalled
```

► 7. Exit from **servera**.

```
[root@servera ~]# exit  
logout  
[student@servera ~]$ exit  
logout  
Connection to servera closed.  
[student@workstation ~]$
```

Finish

On **workstation**, run the **lab software-module finish** script to finish this exercise. This script removes all the modules installed on **servera** during the exercise.

```
[student@workstation ~]$ lab software-module finish
```

This concludes the guided exercise.

▶ Lab

Installing and Updating Software Packages

Performance Checklist

In this lab, you will manage software repositories and module streams, and install and upgrade packages from those repositories and streams.

Outcomes

You should be able to:

- Manage software repositories and module streams.
- Install and upgrade packages from repositories and streams.
- Install an RPM package.

Before You Begin

Log in to **workstation** as **student** using **student** as the password.

On **workstation**, run the **lab software-review start** command. This script ensures that **serverb** is available. It also downloads any packages required for the lab exercise.

```
[student@workstation ~]$ lab software-review start
```

1. On **serverb** configure a software repository to obtain updates. Name the repository as **errata** and configure the repository in the **/etc/yum.repos.d/errata.repo** file. It should access http://content.example.com/rhel8.2/x86_64/rhcsa-practice/errata. Do not check GPG signatures.
2. On **serverb**, install new package **xsane-gimp** and the Apache HTTP Server module from the **2.4** stream and the **common** profile.
3. For security reasons, **serverb** should not be able to send anything to print. Achieve this by removing the **cups** package. Exit from the **root** account.
4. The start script downloads the **rhcsa-script-1.0.0-1.noarch.rpm** package in the **/home/student** directory on **serverb**.

Confirm that the package **rhcsa-script-1.0.0-1.noarch.rpm** is available on **serverb**. Install the package. You will need to gain superuser privileges to install the package. Verify that the package is installed. Exit from **serverb**.

Evaluation

On **workstation**, run the **lab software-review grade** script to confirm success on this lab.

```
[student@workstation ~]$ lab software-review grade
```

Finish

On **workstation**, run the **lab software-review finish** script to complete this exercise. This script removes the repository and packages created during this exercise.

```
[student@workstation ~]$ lab software-review finish
```

This concludes the lab.

► Solution

Installing and Updating Software Packages

Performance Checklist

In this lab, you will manage software repositories and module streams, and install and upgrade packages from those repositories and streams.

Outcomes

You should be able to:

- Manage software repositories and module streams.
- Install and upgrade packages from repositories and streams.
- Install an RPM package.

Before You Begin

Log in to **workstation** as **student** using **student** as the password.

On **workstation**, run the **lab software-review start** command. This script ensures that **serverb** is available. It also downloads any packages required for the lab exercise.

```
[student@workstation ~]$ lab software-review start
```

1. On **serverb** configure a software repository to obtain updates. Name the repository as **errata** and configure the repository in the **/etc/yum.repos.d/errata.repo** file. It should access http://content.example.com/rhel8.2/x86_64/rhcsa-practice/errata. Do not check GPG signatures.

- 1.1. From **workstation** use the **ssh** command to log in to **serverb** as the **student** user.

```
[student@workstation ~]$ ssh student@serverb  
...output omitted...  
[student@serverb ~]$
```

- 1.2. Use the **sudo -i** command to switch to the **root** user.

```
[student@serverb ~]$ sudo -i  
[sudo] password for student: student  
[root@serverb ~]#
```

- 1.3. Create the file **/etc/yum.repos.d/errata.repo** with the following content:

```
[errata]
name=Red Hat Updates
baseurl=http://content.example.com/rhel8.2/x86_64/rhcsa-practice/errata
enabled=1
gpgcheck=0
```

2. On **serverb**, install new package *xsane-gimp* and the *Apache HTTP Server* module from the **2.4** stream and the **common** profile.

- 2.1. Use the **yum list** command to list the available packages for *xsane-gimp*.

```
[root@serverb ~]# yum list xsane-gimp
Last metadata expiration check: 0:24:30 ago on Thu 07 Mar 2019 03:50:55 PM CET.
Available Packages
xsane-gimp.x86_64      0.999-30.el8      rhel-8.2-for-x86_64-appstream-rpms
```

- 2.2. Install the latest version of the *xsane-gimp* package using the **yum install** command.

```
[root@serverb ~]# yum install xsane-gimp
...output omitted...
Install 59 Packages

Total download size: 53 M
Installed size: 217 M
Is this ok [y/N]: y
...output omitted...
Complete!
[root@serverb ~]#
```

- 2.3. List available modules and streams. Look for the *httpd* module. Use the **yum install** command to install the *httpd* module with the **2.4** stream and the **common** profile.

```
[student@serverb ~]$ yum module list
Name      Stream      Profiles          Summary
...output omitted...
httpd     2.4 [d]    common [d], devel, minimal   Apache HTTP Server
...output omitted...
[root@serverb ~]# yum module install httpd:2.4/common
Install 10 Packages

Total download size: 2.1 M
Installed size: 5.7 M
Is this ok [y/N]: y
...output omitted...
Complete!
[root@serverb ~]#
```

3. For security reasons, **serverb** should not be able to send anything to print. Achieve this by removing the *cups* package. Exit from the **root** account.

- 3.1. Use the **yum list** command to list the installed *cups* package.

```
[root@serverb ~]# yum list cups
Installed Packages
cups.x86_64           1:2.2.6-33.el8          @rhel-8.2-for-x86_64-appstream-rpms
[root@serverb ~]#
```

- 3.2. Use the **yum remove** command to remove the **cups** package.

```
[root@serverb ~]# yum remove cups.x86_64
...output omitted...
Remove 9 Packages

Freed space: 11 M
Is this ok [y/N]: y
...output omitted...
Complete!
```

- 3.3. Exit from the **root** account.

```
[root@serverb ~]# exit
[student@serverb ~]$
```

4. The start script downloads the *rhcsa-script-1.0.0-1.noarch.rpm* package in the **/home/student** directory on **serverb**.

Confirm that the package *rhcsa-script-1.0.0-1.noarch.rpm* is available on **serverb**. Install the package. You will need to gain superuser privileges to install the package. Verify that the package is installed. Exit from **serverb**.

- 4.1. Use the **rpm** command to confirm that the *rhcsa-script-1.0.0-1.noarch.rpm* package is available on **serverb** by viewing the package information.

```
[student@serverb ~]$ rpm -q -p rhcsa-script-1.0.0-1.noarch.rpm -i
Name        : rhcsa-script
Version     : 1.0.0
Release     : 1
Architecture: noarch
Install Date: (not installed)
Group       : System
Size        : 1056
License     : GPL
Signature   : (none)
Source RPM  : rhcsa-script-1.0.0-1.src.rpm
Build Date  : Wed 06 Mar 2019 11:29:46 AM CET
Build Host  : foundation0.ilt.example.com
Relocations : (not relocatable)
Packager    : Snehangshu Karmakar
URL         : http://example.com
Summary     : RHCSA Practice Script
Description  :
A RHCSA practice script.
The package changes the motd.
```

- 4.2. Use the **sudo yum localinstall** command to install the *rhcsa-script-1.0.0-1.noarch.rpm* package. The password is **student**.

```
[student@serverb ~]$ sudo yum localinstall \
rhcsa-script-1.0.0-1.noarch.rpm
[sudo] password for student: student
Last metadata expiration check: 1:31:22 ago on Thu 07 Mar 2019 03:50:55 PM CET.
Dependencies resolved.

=====
 Package           Arch    Version      Repository      Size
 =====
 Installing:
 rhcsa-script     noarch  1.0.0-1    @commandline      7.6 k

 Transaction Summary
 =====
 Install 1 Package

 Total size: 7.6 k
 Installed size: 1.0 k
 Is this ok [y/N]: y
 Downloading Packages:
 Running transaction check
 Transaction check succeeded.
 Running transaction test
 Transaction test succeeded.
 Running transaction
 Preparing          :                               1/1
 Running scriptlet: rhcsa-script-1.0.0-1.noarch 1/1
 Installing        : rhcsa-script-1.0.0-1.noarch 1/1
 Running scriptlet: rhcsa-script-1.0.0-1.noarch 1/1
 Verifying         : rhcsa-script-1.0.0-1.noarch 1/1

 Installed:
 rhcsa-script-1.0.0-1.noarch

 Complete!
```

- 4.3. Use the **rpm** command to verify that the package is installed.

```
[student@serverb ~]$ rpm -q rhcsa-script
rhcsa-script-1.0.0-1.noarch
[student@serverb ~]$
```

- 4.4. Exit from **serverb**

```
[student@serverb ~]$ exit
logout
Connection to serverb closed.
[student@workstation ~]$
```

Evaluation

On **workstation**, run the **lab software-review grade** script to confirm success on this lab.

```
[student@workstation ~]$ lab software-review grade
```

Finish

On **workstation**, run the **lab software-review finish** script to complete this exercise. This script removes the repository and packages created during this exercise.

```
[student@workstation ~]$ lab software-review finish
```

This concludes the lab.

Summary

In this chapter, you learned:

- Red Hat Subscription Management provides tools to entitle machines to product subscriptions, get updates to software packages, and track information about support contracts and subscriptions used by the systems.
- Software is provided as RPM packages, which make it easy to install, upgrade, and uninstall software from the system.
- The **rpm** command can be used to query a local database to provide information about the contents of installed packages and install downloaded package files.
- **yum** is a powerful command-line tool that can be used to install, update, remove, and query software packages.
- Red Hat Enterprise Linux 8 uses Application Streams to provide a single repository to host multiple versions of an application's packages and its dependencies.

Accessing Linux File Systems

Goal

Access, inspect, and use existing file systems on storage attached to a Linux server.

Objectives

- Explain what a block device is, interpret the file names of storage devices, and identify the storage device used by the file system for a particular directory or file.
- Access file systems by attaching them to a directory in the file system hierarchy.
- Search for files on mounted file systems using the **find** and **locate** commands.

Sections

- Identifying File Systems and Devices (and Quiz)
- Mounting and Unmounting File Systems (and Guided Exercise)
- Locating Files on the System (and Guided Exercise)

Lab

Accessing Linux File Systems

Identifying File Systems and Devices

Objectives

After completing this section, you should be able to identify a directory in the file system hierarchy and on which device it is stored.

Storage Management Concepts

Files on a Linux server are accessed through the file-system hierarchy, a single inverted tree of directories. This file system hierarchy is assembled from *file systems* provided by the storage devices available to your system. Each file system is a storage device that has been formatted to store files.

In a sense, the Linux file-system hierarchy presents a collection of file systems on separate storage devices as if it were one set of files on one giant storage device that you can navigate. Much of the time, you do not need to know which storage device a particular file is on, you just need to know the directory that file is in.

Sometimes, however, it can be important. You might need to determine how full a storage device is and what directories in the file-system hierarchy are affected. There might be errors in the logs from a storage device, and you need to know what file systems are at risk. You could just want to create a hard link between two files, and you need to know if they are on the same file system to determine if it is possible.

File Systems and Mount Points

To make the contents of a file system available in the file-system hierarchy, it must be *mounted* on an empty directory. This directory is called a *mount point*. Once mounted, if you use `ls` to list that directory, you will see the contents of the mounted file system, and you can access and use those files normally. Many file systems are automatically mounted as part of the boot process.

If you have only worked with Microsoft Windows drive letters, this is a fundamentally different concept. It is somewhat similar to the NTFS mounted folders feature.

File Systems, Storage, and Block Devices

Low-level access to storage devices in Linux is provided by a special type of file called a *block device*. These block devices must be formatted with a file system before they can be mounted.

Block device files are stored in the `/dev` directory, along with other device files. Device files are created automatically by the operating system. In Red Hat Enterprise Linux, the first SATA/PATA, SAS, SCSI, or USB hard drive detected is called `/dev/sda`, the second is `/dev/sdb`, and so on. These names represent the entire hard drive.

Other types of storage will have other forms of naming.

Block Device Naming

Type of device	Device naming pattern
SATA/SAS/USB-attached storage	<code>/dev/sda, /dev/sdb ...</code>
virtio-blk paravirtualized storage (some virtual machines)	<code>/dev/vda, /dev/vdb ...</code>
NVMe-attached storage (many SSDs)	<code>/dev/nvme0, /dev/nvme1 ...</code>
SD/MMC/eMMC storage (SD cards)	<code>/dev/mmcblk0, /dev/mmcblk1 ...</code>



Note

Many virtual machines use the newer **virtio-scsi** paravirtualized storage that will have `/dev/sd*`-style naming.

Disk Partitions

Normally, you do not make the entire storage device into one file system. Storage devices are typically divided up into smaller chunks called *partitions*.

Partitions allow you to compartmentalize a disk: the various partitions can be formatted with different file systems or used for different purposes. For example, one partition can contain user home directories while another can contain system data and logs. If a user fills up the home directory partition with data, the system partition may still have space available.

Partitions are block devices in their own right. On SATA-attached storage, the first partition on the first disk is `/dev/sda1`. The third partition on the second disk is `/dev/sdb3`, and so on. Paravirtualized storage devices have a similar naming system.

An NVMe-attached SSD device names its partitions differently. In that case, the first partition on the first disk is `/dev/nvme0p1`. The third partition on the second disk is `/dev/nvme1p3`, and so on. SD or MMC cards have a similar naming system.

A long listing of the `/dev/sda1` device file on **host** reveals its special file type as **b**, which stands for block device:

```
[user@host ~]$ ls -l /dev/sda1
brw-rw----. 1 root disk 8, 1 Feb 22 08:00 /dev/sda1
```

Logical Volumes

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 in the volume group is then parceled out to one or more *logical volumes*, which are the functional equivalent of a partition residing on a physical disk.

The LVM system assigns names to volume groups and logical volumes upon creation. LVM creates a directory in `/dev` that matches the group name and then creates a symbolic link within that new directory with the same name as the logical volume. That logical volume file is then available to be mounted. For example, if a volume group is called **myvg** and the logical volume within it is called **mylv**, then the full path name to the logical volume device file is `/dev/myvg/mylv`.

**Note**

The form of logical volume device name mentioned above is actually implemented as a symbolic link to the actual device file used to access it, which might vary between boots. There is another form of logical volume device name linked from files in **/dev/mapper** that are often used, and are also symbolic links to the actual device file.

Examining File Systems

To get an overview of local and remote file system devices and the amount of free space available, run the **df** command. When the **df** command is run without arguments, it reports total disk space, used disk space, free disk space, and the percentage of the total disk space used on all mounted regular file systems. It reports on both local and remote file systems.

The following example displays the file systems and mount points on **host**.

```
[user@host ~]$ df
Filesystem      1K-blocks    Used Available Use% Mounted on
/devtmpfs        912584       0   912584   0% /dev
tmpfs           936516       0   936516   0% /dev/shm
tmpfs           936516   16812   919704   2% /run
tmpfs           936516       0   936516   0% /sys/fs/cgroup
/dev/vda3     8377344 1411332   6966012  17% /
/dev/vda1     1038336 169896   868440  17% /boot
tmpfs          187300       0   187300   0% /run/user/1000
```

The partitioning on the **host** system shows two physical file systems, which are mounted on **/** and **/boot**. 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** reports in KiB (2^{10}), MiB (2^{20}), or GiB (2^{30}), while the **-H** option reports in SI units: KB (10^3), MB (10^6), or GB (10^9). Hard drive manufacturers usually use SI units when advertising their products.

Show a report on the file systems on the **host** system with all units converted to human-readable format:

```
[user@host ~]$ df -h
Filesystem      Size  Used Avail Use% Mounted on
/devtmpfs        892M   0    892M   0% /dev
tmpfs           915M   0    915M   0% /dev/shm
tmpfs           915M   17M   899M   2% /run
tmpfs           915M   0    915M   0% /sys/fs/cgroup
/dev/vda3       8.0G  1.4G   6.7G  17% /
/dev/vda1     1014M 166M   849M  17% /boot
tmpfs          183M   0    183M   0% /run/user/1000
```

For more detailed information about space used by a certain directory tree, use 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 **/usr/share** directory on **host**:

```
[root@host ~]# du /usr/share
...output omitted...
176 /usr/share/smarmontools
184 /usr/share/nano
8 /usr/share/cmake/bash-completion
8 /usr/share/cmake
356676 /usr/share
```

Show a disk usage report in human-readable format for the **/usr/share** directory on **host**:

```
[root@host ~]# du -h /var/log
...output omitted...
176K /usr/share/smarmontools
184K /usr/share/nano
8.0K /usr/share/cmake/bash-completion
8.0K /usr/share/cmake
369M /usr/share
```



References

df(1) and **du(1)** man pages

► Quiz

Identifying File Systems and Devices

Choose the correct answers to the following questions:

- ▶ 1. What is the name of the device file of an entire SATA hard drive in the /dev directory?
 - a. /dev/vda
 - b. /dev/sda1
 - c. /dev/sda
 - d. /dev/vg_install/lv_home
- ▶ 2. Choose the device file name of the third partition on the second SATA hard drive.
 - a. /dev/vda2
 - b. /dev/sda3
 - c. /dev/sdb2
 - d. /dev/sdb3
- ▶ 3. What is the name of the device file for the entire second virtio-blk disk attached to a virtual machine?
 - a. /dev/vda2
 - b. /dev/sda2
 - c. /dev/vdb2
 - d. /dev/vdb
- ▶ 4. Choose the correct name of the device file for the third partition on the second virtio-blk disk attached to a virtual machine?
 - a. /dev/vda3
 - b. /dev/sda3
 - c. /dev/vdb3
 - d. /dev/vda3
- ▶ 5. Which command provides an overview of the file system mount points and the amount of free space available in SI units?
 - a. df
 - b. df -H
 - c. df -h
 - d. du -h

► Solution

Identifying File Systems and Devices

Choose the correct answers to the following questions:

- ▶ 1. What is the name of the device file of an entire SATA hard drive in the /dev directory?
 - a. /dev/vda
 - b. /dev/sda1
 - c. /dev/sda
 - d. /dev/vg_install/lv_home
- ▶ 2. Choose the device file name of the third partition on the second SATA hard drive.
 - a. /dev/vda2
 - b. /dev/sda3
 - c. /dev/sdb2
 - d. /dev/sdb3
- ▶ 3. What is the name of the device file for the entire second virtio-blk disk attached to a virtual machine?
 - a. /dev/vda2
 - b. /dev/sda2
 - c. /dev/vdb2
 - d. /dev/vdb
- ▶ 4. Choose the correct name of the device file for the third partition on the second virtio-blk disk attached to a virtual machine?
 - a. /dev/vda3
 - b. /dev/sda3
 - c. /dev/vdb3
 - d. /dev/vda3
- ▶ 5. Which command provides an overview of the file system mount points and the amount of free space available in SI units?
 - a. df
 - b. df -H
 - c. df -h
 - d. du -h

Mounting and Unmounting File Systems

Objectives

After completing this section, you 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 removable storage device needs to be mounted in order 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 directory to use as the mount point in the file-system hierarchy.

There are two common ways to specify the file system on a disk partition to the **mount** command:

- With the name of the device file in **/dev** containing the file system.
- With the *UUID* written to the file system, a universally-unique identifier.

Mounting a device is relatively simple. You need to identify the device you want to mount, make sure the mount point exists, and mount the device on the mount point.

Identifying the Block Device

A hot-pluggable storage device, whether a hard disk drive (HDD) or solid-state device (SSD) in a server caddy, or a USB storage device, might be plugged into a different port each time they are attached to a system.

Use the **lsblk** command to list the details of a specified block device or all the available devices.

```
[root@host ~]# lsblk
NAME      MAJ:MIN RM  SIZE RO TYPE MOUNTPOINT
vda       253:0    0   12G  0 disk
└─vda1    253:1    0    1G  0 part /boot
└─vda2    253:2    0    1G  0 part [SWAP]
└─vda3    253:3    0   11G  0 part /
vdb       253:16   0   64G  0 disk
└─vdb1    253:17   0   64G  0 part
```

If you know that you just added a 64 GB storage device with one partition, then you can guess from the preceding output that **/dev/vdb1** is the partition that you want to mount.

Mounting by Block Device Name

The following example mounts the file system in the **/dev/vdb1** partition on the directory **/mnt/data**.

```
[root@host ~]# mount /dev/vdb1 /mnt/data
```

To mount a file system, the destination directory must already exist. The `/mnt` directory exists by default and is intended for use as a temporary mount point.

You can use `/mnt` directory, or better yet create a subdirectory of `/mnt` to use as a temporary mount point, unless you have a good reason to mount it in a specific location in the file-system hierarchy.



Important

If the directory acting as mount point is not empty, any files copied to that directory before the file system was mounted are not accessible until the file system is unmounted again.

This approach works fine in the short run. However, the order in which the operating system detects disks can change if devices are added to or removed from the system. This will change the device name associated with that storage device. A better approach would be to mount by some characteristic built into the file system.

Mounting by File-system UUID

One stable identifier that is associated with a file system is its UUID, a very long hexadecimal number that acts as a universally-unique identifier. This UUID is part of the file system and remains the same as long as the file system is not recreated.

The `lsblk -fp` command lists the full path of the device, along with the UUIDs and mount points, as well as the type of file system in the partition. If the file system is not mounted, the mount point will be blank.

```
[root@host ~]# lsblk -fp
  NAME      FSTYPE LABEL UUID                                     MOUNTPOINT
/dev/vda
└─/dev/vda1 xfs   boot 23ea8803-a396-494a-8e95-1538a53b821c /boot
└─/dev/vda2 swap   [SWAP] cdf61ded-534c-4bd6-b458-cab18b1a72ea
└─/dev/vda3 xfs   /     44330f15-2f9d-4745-ae2e-20844f22762d
/dev/vdb
└─/dev/vdb1 xfs   46f543fd-78c9-4526-a857-244811be2d88
```

Mount the file system by the UUID of the file system.

```
[root@host ~]# mount UUID="46f543fd-78c9-4526-a857-244811be2d88" /mnt/data
```

Automatic Mounting of Removable Storage Devices

If you are logged in and using the graphical desktop environment, it will automatically mount any removable storage media when it is inserted.

The removable storage device is mounted at `/run/media/USERNAME/LABEL` where `USERNAME` is the name of the user logged into the graphical environment and `LABEL` is an identifier, often the name given to the file system when it was created if one is available.

Before removing the device, you should unmount it manually.

Unmounting File Systems

The shutdown and reboot procedures unmount all file systems automatically. As part of this process, any file system data cached in memory is flushed to the storage device thus ensuring that the file system suffers no data corruption.



Warning

File system data is often cached in memory. Therefore, in order to avoid corrupting data on the disk, it is essential that you unmount removable drives before unplugging them. The unmount procedure synchronizes data before releasing the drive, ensuring data integrity.

To unmount a file system, the **umount** command expects the mount point as an argument.

```
[root@host ~]# umount /mnt/data
```

Unmounting is not possible if the mounted file system is in use. For the **umount** command to succeed, all processes needs to stop accessing data under the mount point.

In the example below, the **umount** fails because the file system is in use (the shell is using **/mnt/data** as its current working directory), generating an error message.

```
[root@host ~]# cd /mnt/data
[root@host data]# umount /mnt/data
umount: /mnt/data: target is busy.
```

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@host data]# lsof /mnt/data
COMMAND PID USER FD TYPE DEVICE SIZE/OFF NODE NAME
bash 1593 root cwd DIR 253,17 6 128 /mnt/data
lsof 2532 root cwd DIR 253,17 19 128 /mnt/data
lsof 2533 root cwd DIR 253,17 19 128 /mnt/data
```

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@host data]# cd
[root@host ~]# umount /mnt/data
```



Note

A common reason for file systems to fail to unmount is that a Bash shell is using the mount point or a subdirectory as a current working directory. Use the **cd** command to change out of the file system to resolve this problem.



References

lsblk(8), **mount(8)**, **umount(8)**, and **lsof(8)** man pages

► Guided Exercise

Mounting and Unmounting File Systems

In this exercise, you will practice mounting and unmounting file systems.

Outcomes

You should be able to identify and mount a new file system at a specified mount point, then unmount it.

Before You Begin

Log in as the **student** user on **workstation** using **student** as the password.

From **workstation**, run the **lab fs-mount start** command. The command runs a start script that determines if the host, **servera**, is reachable on the network. The script also creates a partition on the second disk attached to **servera**.

```
[student@workstation ~]$ lab fs-mount start
```

- 1. Use the **ssh** command to log in to **servera** as the **student** user.

```
[student@workstation ~]$ ssh student@servera  
...output omitted...  
[student@servera ~]$
```

- 2. A new partition with a file system has been added to the second disk (**/dev/vdb**) on **servera**. Mount the newly available partition by UUID at the newly created mount point **/mnt/newspace**.
- 2.1. Use the **sudo -i** command to switch to **root**, as only the **root** user can manually mount a device.

```
[student@servera ~]$ sudo -i  
[sudo] password for student: student  
[root@servera ~]#
```

- 2.2. Create the **/mnt/newspace** directory.

```
[root@servera ~]# mkdir /mnt/newspace
```

- 2.3. Use the **lsblk** command with the **-fp** option to discover the UUID of the device, **/dev/vdb1**.

```
[root@servera ~]# lsblk -fp /dev/vdb
NAME      FSTYPE LABEL UUID                                     MOUNTPOINT
/dev/vdb
└─/dev/vdb1 xfs    a04c511a-b805-4ec2-981f-42d190fc9a65
```

- 2.4. Mount the file system by using UUID on the **/mnt/newspace** directory. Replace the UUID with that of the **/dev/vdb1** disk from the previous command output.

```
[root@servera ~]# mount \
UUID="a04c511a-b805-4ec2-981f-42d190fc9a65" /mnt/newspace
```

- 2.5. Verify that the **/dev/vdb1** device is mounted on the **/mnt/newspace** directory.

```
[root@servera ~]# lsblk -fp /dev/vdb
NAME      FSTYPE LABEL UUID                                     MOUNTPOINT
/dev/vdb
└─/dev/vdb1 xfs    a04c511a-b805-4ec2-981f-42d190fc9a65 /mnt/newspace
```

- ▶ 3. Change to the **/mnt/newspace** directory and create a new directory, **/mnt/newspace/newdir**, with an empty file, **/mnt/newspace/newdir/newfile**.

- 3.1. Change to the **/mnt/newspace** directory.

```
[root@servera ~]# cd /mnt/newspace
```

- 3.2. Create a new directory, **/mnt/newspace/newdir**.

```
[root@servera newspace]# mkdir newdir
```

- 3.3. Create a new empty file, **/mnt/newspace/newdir/newfile**.

```
[root@servera newspace]# touch newdir/newfile
```

- ▶ 4. Unmount the file system mounted on the **/mnt/newspace** directory.

- 4.1. Use the **umount** command to unmount **/mnt/newspace** while the current directory on the shell is still **/mnt/newspace**. The **umount** command fails to unmount the device.

```
[root@servera newspace]# umount /mnt/newspace
umount: /mnt/newspace: target is busy.
```

- 4.2. Change the current directory on the shell to **/root**.

```
[root@servera newspace]# cd
[root@servera ~]#
```

- 4.3. Now, successfully unmount **/mnt/newspace**.

```
[root@servera ~]# umount /mnt/newspace
```

► 5. Exit from **servera**.

```
[root@servera ~]# exit  
logout  
[student@servera ~]$ exit  
logout  
Connection to servera closed.  
[student@workstation]$
```

Finish

On **workstation**, run the **lab fs-mount finish** script to complete this exercise.

```
[student@workstation ~]$ lab fs-mount finish
```

This concludes the guided exercise.

Locating Files on the System

Objectives

After completing this section, you should be able to search for files on mounted file systems using **find** and **locate**.

Searching for Files

A system administrator needs tools to search for files matching certain criteria on the file system. This section discusses two commands that can search for files in the file-system hierarchy.

- The **locate** command searches a pregenerated index for file names or file paths and returns the results instantly.
- The **find** command searches for files in real time by crawling through the file-system hierarchy.

Locating Files by Name

The **locate** command finds files based on the name or path to the file. It is fast because it looks up this information from the **mlocate** database. However, this database is not updated in real time, and it must be frequently updated for results to be accurate. This also means that **locate** will not find files that have been created since the last update of the database.

The **locate** database is automatically updated every day. However, at any time the **root** user can issue the **updatedb** command to force an immediate update.

```
[root@host ~]# updatedb
```

The **locate** command restricts results for unprivileged users. In order to see the resulting file name, the user must have search permission on the directory in which the file resides.

Search for files with **passwd** in the name or path in directory trees readable by **user** on **host**.

```
[user@host ~]$ 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
...output omitted...
```

Results are returned even when the file name or path is only a partial match to the search query.

```
[root@host ~]# locate image
/etc/selinux/targeted-contexts/virtual_image_context
/usr/bin/grub2-mkimage
/usr/lib/sysimage
/usr/lib/dracut/dracut.conf.d/02-generic-image.conf
/usr/lib/firewalld/services/ovirt-imageio.xml
/usr/lib/grub/i386-pc/lnxboot.image
...output omitted...
```

The **-i** option performs a case-insensitive search. With this option, all possible combinations of upper and lowercase letters match the search.

```
[user@host ~]$ locate -i messages
...output omitted...
/usr/share/vim/vim80/lang/zh_TW/LC_MESSAGES
/usr/share/vim/vim80/lang/zh_TW/LC_MESSAGES/vim.mo
/usr/share/vim/vim80/lang/zh_TW.UTF-8/LC_MESSAGES
/usr/share/vim/vim80/lang/zh_TW.UTF-8/LC_MESSAGES/vim.mo
/usr/share/vim/vim80/syntax/messages.vim
/usr/share/vim/vim80/syntax/msmessages.vim
/var/log/messages
```

The **-n** option limits the number of returned search results by the **locate** command. The following example limits the search results returned by **locate** to the first five matches:

```
[user@host ~]$ 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
```

Searching for Files in Real Time

The **find** command locates files by performing a real-time search in the file-system hierarchy. It is slower than **locate**, but more accurate. It can also search for files based on criteria other than the file name, such as the permissions of the file, type of file, its size, or its modification time.

The **find** command looks at files in the file system using the user account that executed the search. 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** starts the search in the current directory and looks for matches in any subdirectory.

To search for files by file name, use the **-name FILENAME** option. With this option, **find** returns the path to files matching *FILENAME* exactly. For example, to search for files named **sshd_config** starting from the / directory, run the following command:

```
[root@host ~]# find / -name sshd_config
/etc/ssh/sshd_config
```

**Note**

With the **find** command, the full word options use a single dash and options follow the path name argument, unlike most other Linux commands.

Wildcards are available to search for a file name and return all results that are a partial match. When using wildcards, it is important to quote the file name to look for to prevent the terminal from interpreting the wildcard.

In the following example, search for files starting in the `/` directory that end in `.txt`:

```
[root@host ~]# 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
...output omitted...
```

To search for files in the `/etc/` directory that contain the word, **pass**, anywhere in their names on **host**, run the following command:

```
[root@host ~]# find /etc -name '*pass*'
/etc/security/opasswd
/etc/pam.d/passwd
/etc/pam.d/password-auth
/etc/passwd-
/etc/passwd
/etc/authselect/password-auth
```

To perform a case-insensitive search for a given file name, use the **-iname** option, followed by the file name to search. To search files with case-insensitive text, **messages**, in their names in the `/` directory on **host**, run the following command:

```
[root@host ~]# find / -iname '*messages*'
...output omitted...
/usr/share/vim/vim80/lang/zh_CN.UTF-8/LC_MESSAGES
/usr/share/vim/vim80/lang/zh_CN.cp936/LC_MESSAGES
/usr/share/vim/vim80/lang/zh_TW/LC_MESSAGES
/usr/share/vim/vim80/lang/zh_TW.UTF-8/LC_MESSAGES
/usr/share/vim/vim80/syntax/messages.vim
/usr/share/vim/vim80/syntax/msmessages.vim
```

Searching Files Based on Ownership or Permission

The **find** command 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 **user** in the `/home/user` directory on **host**.

```
[user@host ~]$ find -user user
.
./.bash_logout
./.bash_profile
./.bashrc
./.bash_history
```

Search for files owned by the group **user** in the **/home/user** directory on **host**.

```
[user@host ~]$ find -group user
.
./.bash_logout
./.bash_profile
./.bashrc
./.bash_history
```

Search for files owned by user ID **1000** in the **/home/user** directory on **host**.

```
[user@host ~]$ find -uid 1000
.
./.bash_logout
./.bash_profile
./.bashrc
./.bash_history
```

Search for files owned by group ID **1000** in the **/home/user** directory on **host**.

```
[user@host ~]$ find -gid 1000
.
./.bash_logout
./.bash_profile
./.bashrc
./.bash_history
```

The **-user**, and **-group** options can be used together to search files where file owner and group owner are different. The example below list files that are both owned by user **root** and affiliated with group **mail**.

```
[root@host ~]# find / -user root -group mail
/var/spool/mail
...output omitted...
```

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 **/** matches 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 matches 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@host ~]# 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@host ~]# 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@host ~]# find /home -perm /442
```

When used with `/` or `-`, a value of `0` works like a wildcard, since it means *a permission of at least nothing*.

To match any file in the `/home/user` directory for which others have at least read access on `host`, run:

```
[user@host ~]$ find -perm -004
```

Find all files in the `/home/user` directory where `other` has write permissions on `host`.

```
[user@host ~]$ find -perm -002
```

Searching Files Based on Size

The `find` command can look up files that match a size specified with the `-size` option, followed by a numerical value and the unit. Use the following list as the units with the `-size` option:

- **k**, for kilobyte
- **M**, for megabyte
- **G**, for gigabyte

The example below shows how to search for files with a size of 10 megabytes, rounded up.

```
[user@host ~]$ find -size 10M
```

To search the files with a size *more than* 10 gigabytes.

```
[user@host ~]$ find -size +10G
```

To list all files with a size *less than* 10 kilobytes.

```
[user@host ~]$ find -size -10k
```

**Important**

The **-size** option unit modifiers round everything up to single units. For example, the **find -size 1M** command shows files smaller than 1 MB because it rounds all files up to 1 MB.

Searching Files Based on Modification Time

The **-mmin** option, followed by the time in minutes, searches for all files that had their content changed at **n** minutes ago in the past. The file's timestamp is always rounded down. It also supports fractional values when used with ranges (**+n** and **-n**).

To find all files that had their file content changed 120 minutes ago on **host**, run:

```
[root@host ~]# find / -mmin 120
```

The **+** modifier in front of the amount of minutes looks for all files in the **/** that have been modified more than **n** minutes ago. In this example, files that were modified more than 200 minutes ago are listed.

```
[root@host ~]# find / -mmin +200
```

The **-** modifier changes the search to look for all files in the **/** directory which have been changed less than **n** minutes ago. In this example, files that were modified less than 150 minutes ago are listed.

```
[root@host ~]# find / -mmin -150
```

Searching Files Based on File Type

The **-type** option in the **find** command limits the search scope to a given file type. Use the following list to pass the required flags to limit the scope of search:

- **f**, for regular file
- **d**, for directory
- **l**, for soft link
- **b**, for block device

Search for all directories in the **/etc** directory on **host**.

```
[root@host ~]# find /etc -type d
/etc
/etc/tmpfiles.d
/etc/systemd
/etc/systemd/system
/etc/systemd/system/getty.target.wants
...output omitted...
```

Search for all soft links on **host**.

```
[root@host ~]# find / -type l
```

Generate a list of all block devices in the `/dev` directory on **host**:

```
[root@host ~]# 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 **host**:

```
[root@host ~]# find / -type f -links +1
```



References

`locate(1)`, `updatedb(8)`, and `find(1)` man pages

► Guided Exercise

Locating Files on the System

In this exercise, you will find specific files on mounted file systems by using the **find** and **locate** commands.

Outcomes

You should be able search files using the **find** and **locate** commands.

Before You Begin

Log in as the **student** user on **workstation** using **student** as the password.

From **workstation**, run the **lab fs-locate start** command. The command runs a start script that determines whether the host, **servera**, is reachable on the network.

```
[student@workstation ~]$ lab fs-locate start
```

- 1. Use the **ssh** command to log in to **servera** as the **student** user.

```
[student@workstation ~]$ ssh student@servera  
...output omitted...  
[student@servera ~]$
```

- 2. Use the **locate** command to search files on **servera**.

- 2.1. Even though the **locate** database is updated automatically every day, make sure that the database is up-to-date by manually starting an update on **servera**. Use the **sudo updatedb** command to update the database used by the **locate** command.

```
[student@servera ~]$ sudo updatedb  
[sudo] password for student: student  
[student@servera ~]$
```

- 2.2. Locate the **logrotate.conf** configuration file.

```
[student@servera ~]$ locate logrotate.conf  
/etc/logrotate.conf  
/usr/share/man/man5/logrotate.conf.5.gz
```

- 2.3. Locate the **networkmanager.conf** configuration file, ignoring case.

```
[student@servera ~]$ 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
```

- 3. Use the **find** command to perform real-time searches on **servera** according to the following requirements:
- Search all files in the **/var/lib** directory that are owned by the **chrony** user.
 - List all files in the **/var** directory that are owned by **root** and the group owner is **mail**.
 - List all files in the **/usr/bin** directory that has a file size greater than 50 KB.
 - Search all files in the **/home/student** directory that have not been changed in the last 120 minutes.
 - List all the block device files in the **/dev** directory.
- 3.1. Use the **find** command to search all files in the **/var/lib** directory those are owned by the **chrony** user. Use the **sudo** command as the files inside the **/var/lib** directory are owned by **root**.

```
[student@servera ~]$ sudo find /var/lib -user chrony  
[sudo] password for student: student  
/var/lib/chrony  
/var/lib/chrony/drift
```

- 3.2. List all files in the **/var** directory that are owned by **root** and are affiliated with the **mail** group.

```
[student@servera ~]$ sudo find /var -user root -group mail  
/var/spool/mail
```

- 3.3. List all files in the **/usr/bin** directory with a file size greater than 50 KB.

```
[student@servera ~]$ find /usr/bin -size +50k  
/usr/bin/iconv  
/usr/bin/locale  
/usr/bin/localedef  
/usr/bin/cmp  
...output omitted...
```

- 3.4. Find all files in the **/home/student** directory that have not been changed in the last 120 minutes.

```
[student@servera ~]$ find /home/student -mmin +120  
/home/student/.bash_logout  
/home/student/.bash_profile  
/home/student/.bashrc  
...output omitted...
```

- 3.5. List all block device files in the **/dev** directory.

```
[student@servera ~]$ find /dev -type b
/dev/vdd
/dev/vdc
/dev/vdb
/dev/vda3
/dev/vda2
/dev/vda1
/dev/vda
```

► 4. Exit from **servera**.

```
[student@servera ~]$ exit
logout
Connection to servera closed.
[student@workstation]$
```

Finish

On **workstation**, run the **lab fs-locate finish** script to complete this exercise.

```
[student@workstation ~]$ lab fs-locate finish
```

This concludes the guided exercise.

▶ Lab

Accessing Linux File Systems

Performance Checklist

In this lab, you will mount a local file system and locate specific files on that file system.

Outcomes

You should be able to:

- Mount a file system.
- Generate a disk usage report.
- Search files in the local file system.

Before You Begin

Log in as the **student** user on **workstation** using **student** as the password.

From **workstation**, run the **lab fs-review start** command. The command runs a start script that determines if the host, **serverb**, is reachable on the network. The script also creates a partition on the second disk attached to **serverb**.

```
[student@workstation ~]$ lab fs-review start
```

1. On **serverb** as **root**, identify the UUID for **/dev/vdb1** and mount **/dev/vdb1** by its UUID on the **/mnt/freespace** directory.
2. Generate a disk usage report of the **/usr/share** directory, and save the result in the **/mnt/freespace/results.txt** file.
3. Use the **locate** command to find all **rsyslog.conf** configuration files and store the result in the **/mnt/freespace/search1.txt** file.
4. Store the search result of all files in the **/usr/share** directory that is greater than 50 MB and less than 100 MB in the **/mnt/freespace/search2.txt** file.
5. Exit from **serverb**.

Evaluation

On **workstation**, run the **lab fs-review grade** script to confirm success on this lab.

```
[student@workstation ~]$ lab fs-review grade
```

Finish

On **workstation**, run the **lab fs-review finish** script to complete this exercise.

```
[student@workstation ~]$ lab fs-review finish
```

This concludes the lab.

► Solution

Accessing Linux File Systems

Performance Checklist

In this lab, you will mount a local file system and locate specific files on that file system.

Outcomes

You should be able to:

- Mount a file system.
- Generate a disk usage report.
- Search files in the local file system.

Before You Begin

Log in as the **student** user on **workstation** using **student** as the password.

From **workstation**, run the **lab fs-review start** command. The command runs a start script that determines if the host, **serverb**, is reachable on the network. The script also creates a partition on the second disk attached to **serverb**.

```
[student@workstation ~]$ lab fs-review start
```

1. On **serverb** as **root**, identify the UUID for **/dev/vdb1** and mount **/dev/vdb1** by its UUID on the **/mnt/freespace** directory.

- 1.1. Use the **ssh** command to log in to **serverb** as the **student** user.

```
[student@workstation ~]$ ssh student@serverb
...output omitted...
[student@serverb ~]$
```

- 1.2. Use the **su -** command to switch to **root**.

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

- 1.3. Use the **lsblk** command to determine the UUID of the **/dev/vdb1** device.

NAME	FSTYPE	LABEL	UUID	MOUNTPOINT
/dev/vdb				
└─/dev/vdb1	xfs		a04c511a-b805-4ec2-981f-42d190fc9a65	

- 1.4. Create the **/mnt/freespace** directory.

```
[root@serverb ~]# mkdir /mnt/freespace
```

- 1.5. Mount the **/dev/vdb1** device by using the UUID on the **/mnt/freespace** directory.

```
[root@serverb ~]# mount UUID="a04c511a-b805-4ec2-981f-42d190fc9a65" /mnt/freespace
```

- 1.6. Verify that the **/dev/vdb1** device is mounted on the **/mnt/freespace** directory.

NAME	FSTYPE	LABEL	UUID	MOUNTPOINT
/dev/vdb				
└─/dev/vdb1	xfs		a04c511a-b805-4ec2-981f-42d190fc9a65	/mnt/freespace

2. Generate a disk usage report of the **/usr/share** directory, and save the result in the **/mnt/freespace/results.txt** file.

```
[root@serverb ~]# du /usr/share > /mnt/freespace/results.txt
```

3. Use the **locate** command to find all **rsyslog.conf** configuration files and store the result in the **/mnt/freespace/search1.txt** file.

- 3.1. Use the **updatedb** command to update the database used by **locate**.

```
[root@serverb ~]# updatedb
```

- 3.2. Locate **rsyslog.conf** configuration files and save the result in the **/mnt/freespace/search1.txt** file.

```
[root@serverb ~]# locate rsyslog.conf > /mnt/freespace/search1.txt
```

4. Store the search result of all files in the **/usr/share** directory that is greater than 50 MB and less than 100 MB in the **/mnt/freespace/search2.txt** file.

```
[root@serverb ~]# find /usr/share -size +50M -size -100M > \
/mnt/freespace/search2.txt
```

5. Exit from **serverb**.

```
[root@serverb ~]$ exit
logout
[student@serverb ~]$ exit
logout
Connection to serverb closed.
[student@workstation]$
```

Evaluation

On **workstation**, run the **lab fs-review grade** script to confirm success on this lab.

```
[student@workstation ~]$ lab fs-review grade
```

Finish

On **workstation**, run the **lab fs-review finish** script to complete this exercise.

```
[student@workstation ~]$ lab fs-review finish
```

This concludes the lab.

Summary

In this chapter, you learned:

- Storage devices are represented by a special file type called block device.
- The **df** command reports total disk space, used disk space, and free disk space on all mounted regular file systems.
- The **mount** command allows the **root** user to manually mount a file system.
- All processes need to stop accessing the mount point in order to successfully unmount the device.
- The removable storage devices are mounted in the **/run/media** directory when using the graphical environment.
- The **find** command performs a real-time search in the local file systems to find files based on search criteria.

Analyzing Servers and Getting Support

Goal

Investigate and resolve issues in the web-based management interface, getting support from Red Hat to help solve problems.

Objectives

- Activate the Web Console management interface to remotely manage and monitor the performance of a Red Hat Enterprise Linux server.
- Describe key resources available through the Red Hat Customer Portal, and find information from Red Hat documentation and the Knowledgebase.
- Analyze servers for issues, remediate or resolve them, and confirm the solution with Red Hat Insights.

Sections

- Analyzing and Managing Remote Servers (and Guided Exercise)
- Getting Help from Red Hat Customer Portal (and Guided Exercise)
- Detecting and Resolving Issues with Red Hat Insights (and Quiz)

Analyzing and Managing Remote Servers

Objectives

After completing this section, you should be able to activate the Web Console management interface to remotely manage and monitor the performance of a Red Hat Enterprise Linux server.

Describing the Web Console

Web Console is a web-based management interface for Red Hat Enterprise Linux 8 designed for managing and monitoring your servers. It is based on the open source Cockpit service.

You can use Web Console to monitor system logs and view graphs of system performance. Additionally, you can use your web browser to change settings using graphical tools in the Web Console interface, including a fully-functional interactive terminal session.

Enabling the Web Console

Red Hat Enterprise Linux 8 installs Web Console by default in all installation variants except a minimal installation. Use the following command to install Web Console:

```
[user@host ~]$ sudo yum install cockpit
```

Enable and start the **cockpit.socket** service, which runs a web server. This step is necessary if you need to connect to the system through the web interface.

```
[user@host ~]$ sudo systemctl enable --now cockpit.socket
Created symlink /etc/systemd/system/sockets.target.wants/cockpit.socket -> /usr/
lib/systemd/system/cockpit.socket.
```

If you are using a custom firewall profile, you need to add the **cockpit** service to **firewalld** to open port 9090 in the firewall:

```
[user@host ~]$ sudo firewall-cmd --add-service=cockpit --permanent
success
[user@host ~]$ sudo firewall-cmd --reload
success
```

Logging in to the Web Console

Web Console provides its own web server. Launch Firefox to log in to Web Console. You can log in with the user name and password of any local account on the system, including the **root** user.

Open `https://servername:9090` in your web browser, where `servername` is the host name or IP address of your server. The connection will be protected by a TLS session. The system is installed with a self-signed TLS certificate by default, and when you initially connect your web browser will probably display a security warning. The **cockpit-ws(8)** man page provides instructions on how to replace the TLS certificate with one that is properly signed.

Enter your user name and password at the login screen.

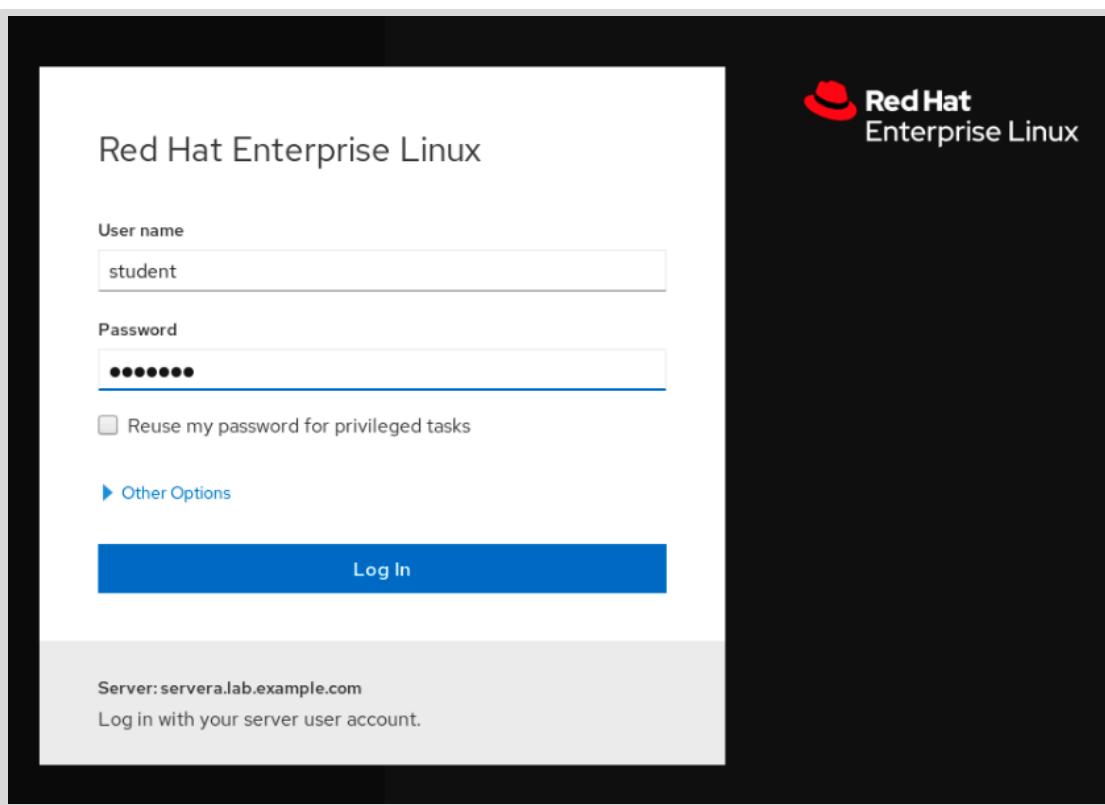


Figure 16.1: The Web Console login screen

Optionally, click the **Reuse my password for privileged tasks** option. This permits you to execute commands with sudo privileges, allowing you to perform tasks such as modifying system information or configuring new accounts.

Click **Log In**.

Web Console displays the user name on the right side of the title bar. If you choose the **Reuse my password for privileged tasks** option, the **Privileged** icon displays to the left of the user name.

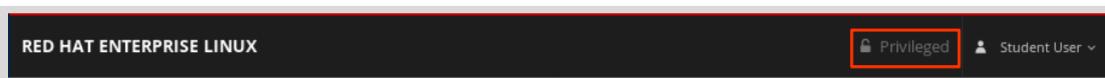


Figure 16.2: Privileged user's title bar

If you are logged in as a non-privileged user, the **Privileged** icon is not displayed.

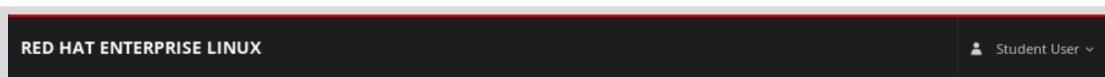


Figure 16.3: Non-privileged user's title bar

Changing Passwords

Privileged and non-privileged users can change their own passwords while logged in to Web Console. Click **Accounts** on the navigation bar. Click your account label to open the account details page.

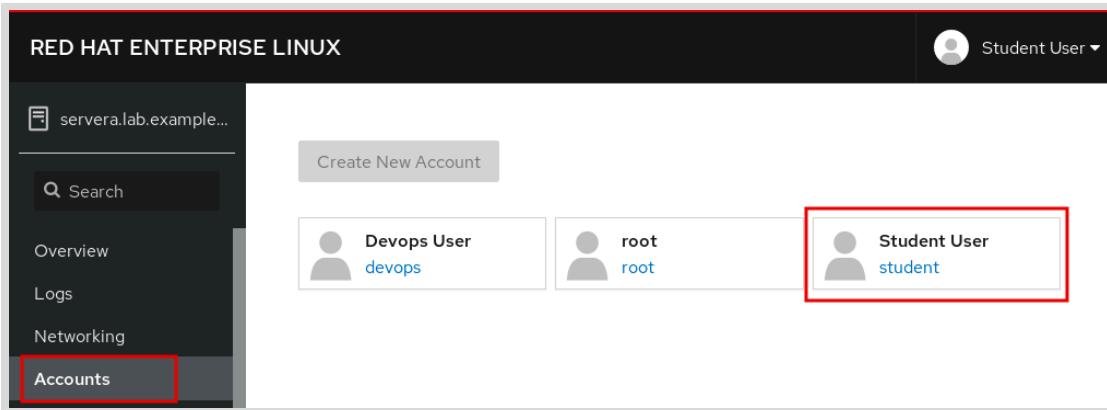


Figure 16.4: Displaying user accounts

As a non-privileged user, you are restricted to setting or resetting your password and managing public SSH keys. To set or reset your password, click **Set Password**.

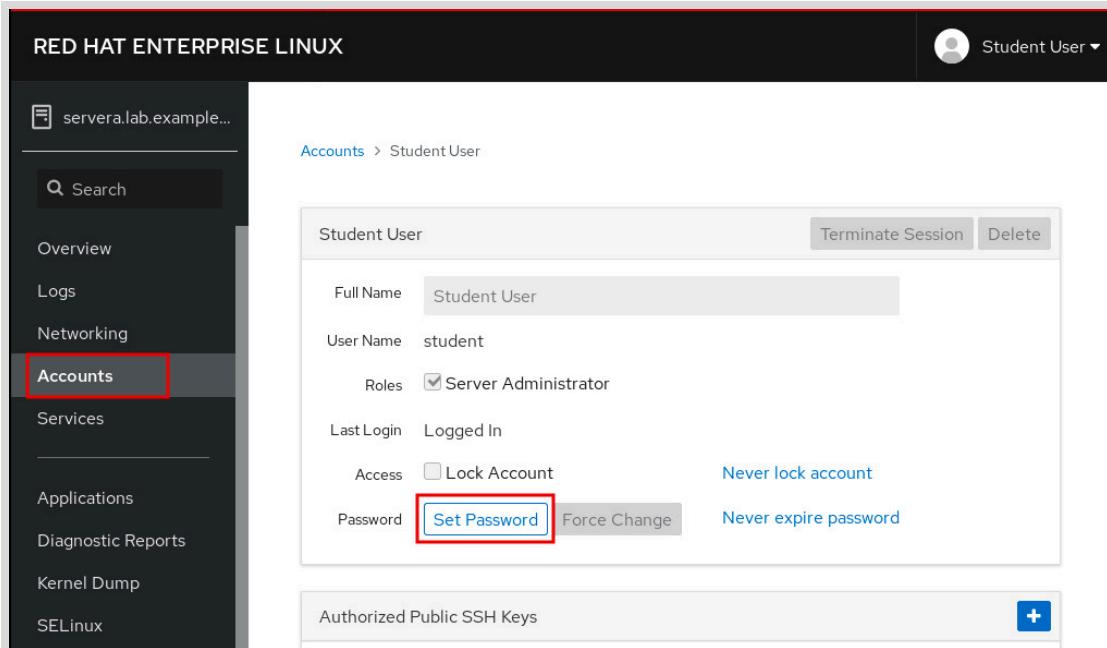


Figure 16.5: User account details

Enter your information in the **Old Password**, **New Password**, and **Confirm New Password** fields. Click **Set** to activate the new password.

Figure 16.6: Setting and resetting passwords

Troubleshooting with the Web Console

Web Console is a powerful troubleshooting tool. You can monitor basic system statistics in real time, inspect system logs, and quickly switch to a terminal session within Web Console to gather additional information from the command-line interface.

Monitoring System Statistics in Real Time

Click **Overview** on the navigation bar to view information about the system, such as its type of hardware, operating system, host name, and more. Notice that if you are logged in as a non-privileged user, you see all the information but you are not permitted to modify values. The following image displays part of the **Overview** page.

Figure 16.7: Non-privileged user's Overview page

Click **View graphs** on the **Overview** page to view graphs of current system performance for CPU activity, memory use, disk I/O, and network utilization.

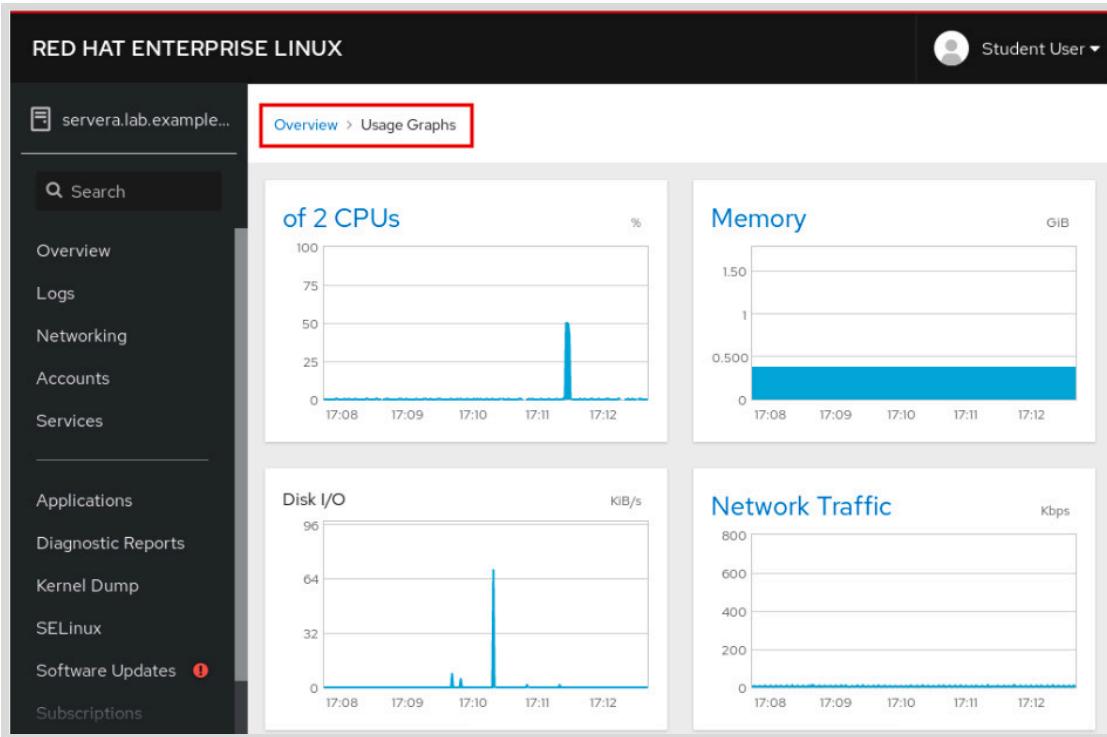


Figure 16.8: Non-privileged user's system performance metrics

Inspecting and Filtering Syslog Events

Logs in the navigation bar provides access to analysis tools for the system logs. You can use the menus on the page to filter log messages based on a logging date range, severity level, or both. Web Console uses the current date as the default, but you can click the date menu and specify any range of dates. Similarly, the **Severity** menu provides options ranging from **Everything** to more specific severity conditions such as **Alert and above**, **Debug and above**, and so on.

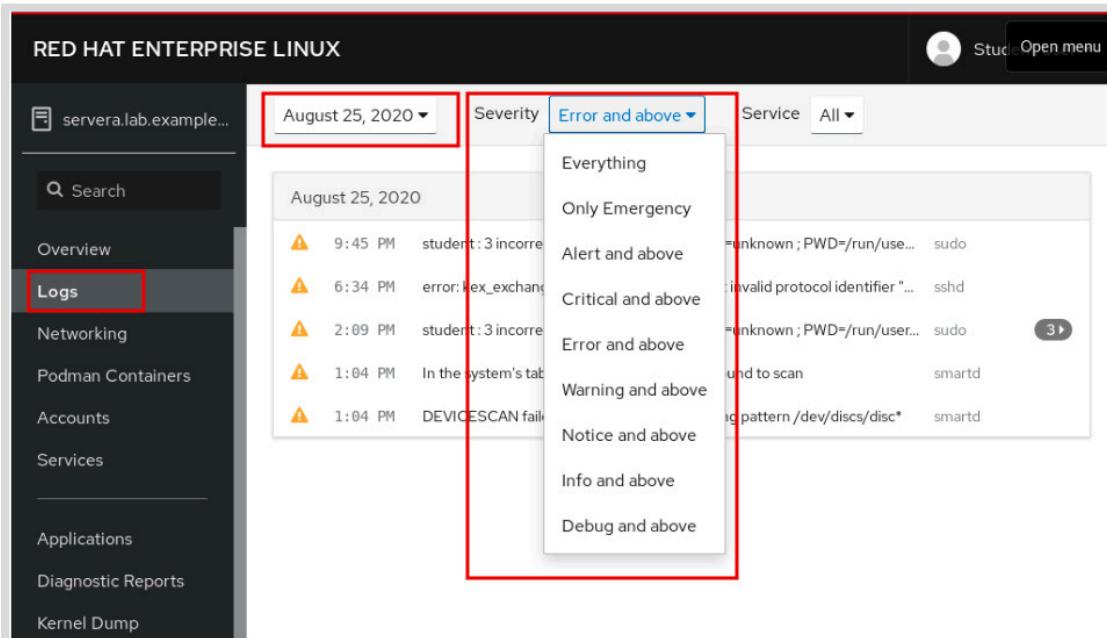


Figure 16.9: Log severity selections

Click a row to view details of the log report. In the example below, note the first row that reports on a **sudo** log message.

The screenshot shows the Red Hat Enterprise Linux cockpit interface. On the left, a sidebar menu includes 'Logs' (which is highlighted with a red box), 'Overview', 'Networking', 'Podman Containers', 'Accounts', 'Services', 'Applications', 'Diagnostic Reports', 'Kernel Dump', and 'SELinux'. The main content area displays a log viewer for August 26, 2020. One log entry is selected, showing the date (August 26, 2020), time (7:41 AM), priority (Warning), facility (sudo), and message (student : 3 incorrect password attempts; TTY=unknown; PWD=/run/user/1000; USER=root; COMMAND=/bin/cockpit-bridge --privileged). Other log entries listed include 'smartd' and 'smartd'.

Figure 16.10: Log entry selection

The example below shows the details displayed when you click the **sudo** row. Details of the report include the selected log entry (**sudo**), the date, time, priority, and syslog facility of the log entry, the host name of the system that reported the log message, and more.

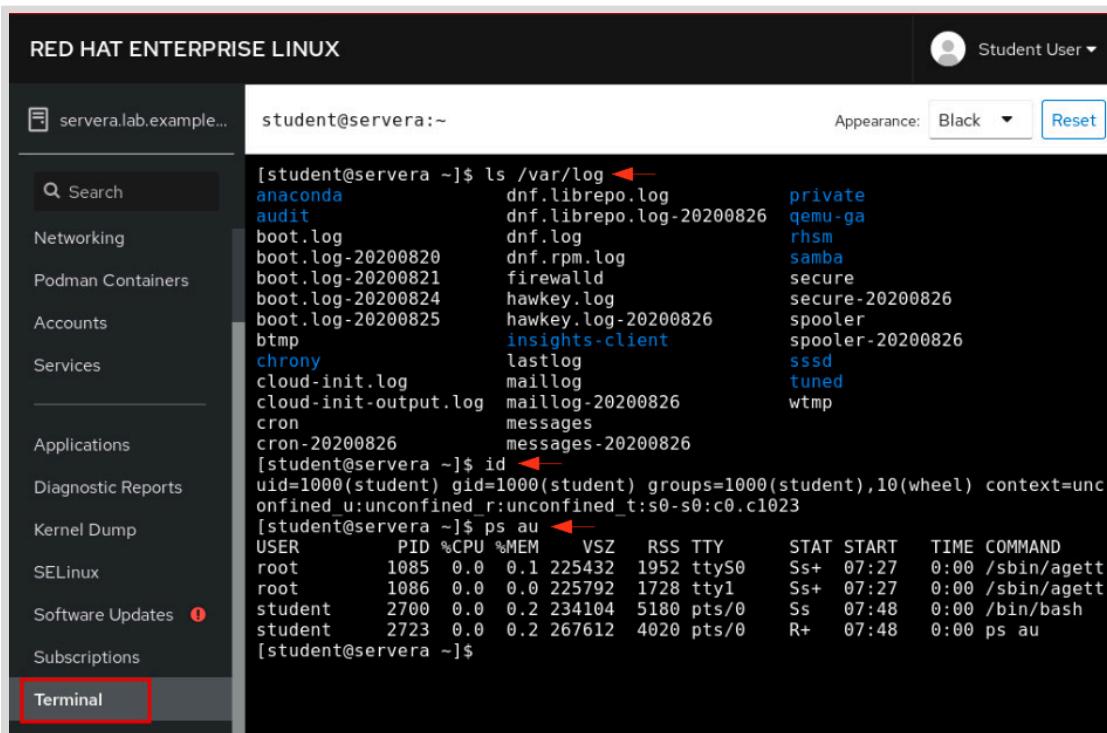
The screenshot shows the Red Hat Enterprise Linux cockpit interface. The 'Logs' menu item is highlighted with a red box. The main content area shows the details of a selected log entry. The log entry is titled 'sudo' and includes the following fields: student : 3 incorrect password attempts; TTY=unknown; PWD=/run/user/1000; USER=root; COMMAND=/bin/cockpit-bridge --privileged. Below this, various metadata fields are listed: PRIORITY (1), SYSLOG_FACILITY (10), SYSLOG_IDENTIFIER (sudo), _AUDIT_LOGINUID (1000), _AUDIT_SESSION (3), _BOOT_ID (d8e459d758dc41c099c917b21e9e51ed), _CAP_EFFECTIVE (3fffffff), _COMM (sudo), _GID (1000), _HOSTNAME (servera.lab.example.com), and _MACHINE_ID (f874df04639f474cb0a9881041f4f7d4).

Figure 16.11: Log entry details

Running Commands from a Terminal Session

Terminal in the navigation bar provides access to a fully-functional terminal session within the Web Console interface. This allows you to run arbitrary commands to manage and work with the system and to perform tasks not supported by the other tools provided by Web Console.

The following image displays examples of common commands used to gather additional information. Listing the contents of the **/var/log** directory provides reminders of log files that may have valuable information. The **id** command provides quick information such as group membership that may help troubleshoot file access restrictions. The **ps au** command provides a quick view of processes running in the terminal and the user associated with the process.



```
[student@servera ~]$ ls /var/log
anaconda      dnf.librepo.log    private
audit         dnf.librepo.log-20200826  qemu-ga
boot.log      dnf.log           rhsm
boot.log-20200820   dnf.rpm.log    samba
boot.log-20200821   firewalld     secure
boot.log-20200824   hawkey.log    secure-20200826
boot.log-20200825   hawkey.log-20200826 spooler
btmp          insights-client   spooler-20200826
chrony        lastlog         sssd
cloud-init.log maillog         tuned
cloud-init-output.log maillog-20200826 wtmp
cron          messages        cron-20200826
cron-20200826   messages-20200826
[student@servera ~]$ id
uid=1000(student) gid=1000(student) groups=1000(student),10(wheel) context=unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023
[student@servera ~]$ ps au
USER      PID %CPU %MEM   VSZ   RSS TTY      STAT START   TIME COMMAND
root      1085  0.0  0.1 225432  1952  ttyS0    Ss+  07:27  0:00 /sbin/agetty
root      1086  0.0  0.0 225792  1728  tty1    Ss+  07:27  0:00 /sbin/agetty
student   2700  0.0  0.2 234104  5180  pts/0    Ss  07:48  0:00 /bin/bash
student   2723  0.0  0.2 267612  4020  pts/0    R+  07:48  0:00 ps au
[student@servera ~]$
```

Figure 16.12: Non-privileged terminal session troubleshooting

Creating Diagnostic Reports

A diagnostic report is a collection of configuration details, system information, and diagnostic information from a Red Hat Enterprise Linux system. Data collected in the completed report includes system logs and debug information that can be used to troubleshoot issues.

Log in to Web Console as a privileged user. Click **Diagnostic Reports** on the navigation bar to open the page that creates these reports. Click **Create Report** to generate a new diagnostic report.

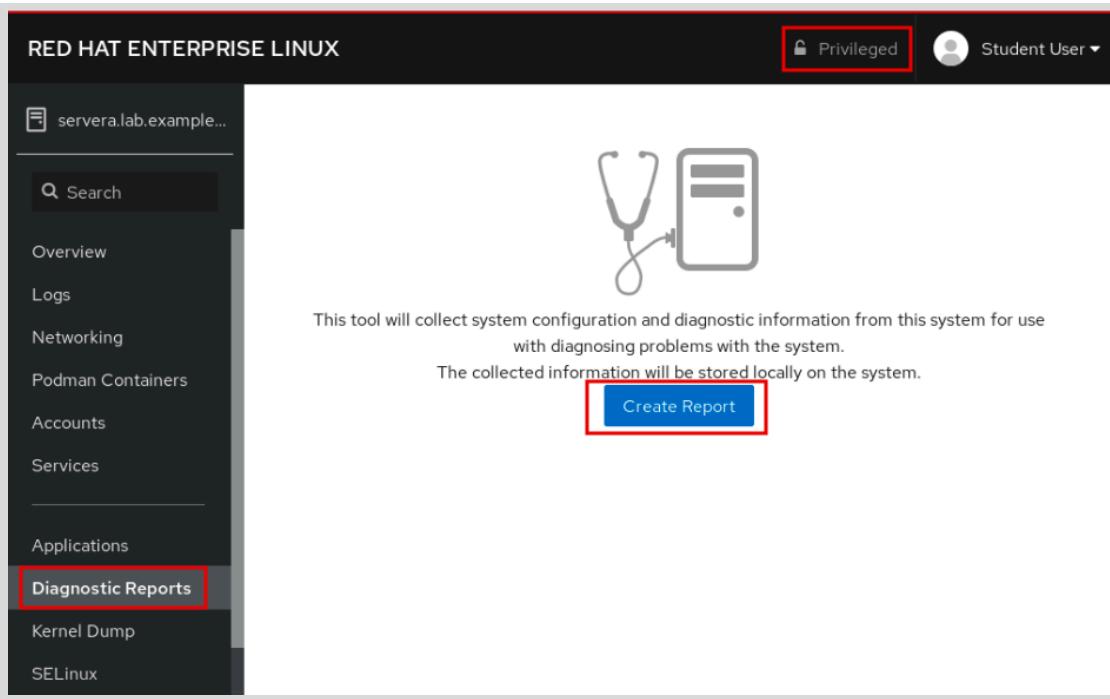


Figure 16.13: Creating a diagnostic report

The interface displays **Done!** when the report is complete. Click **Download report** to save the report.

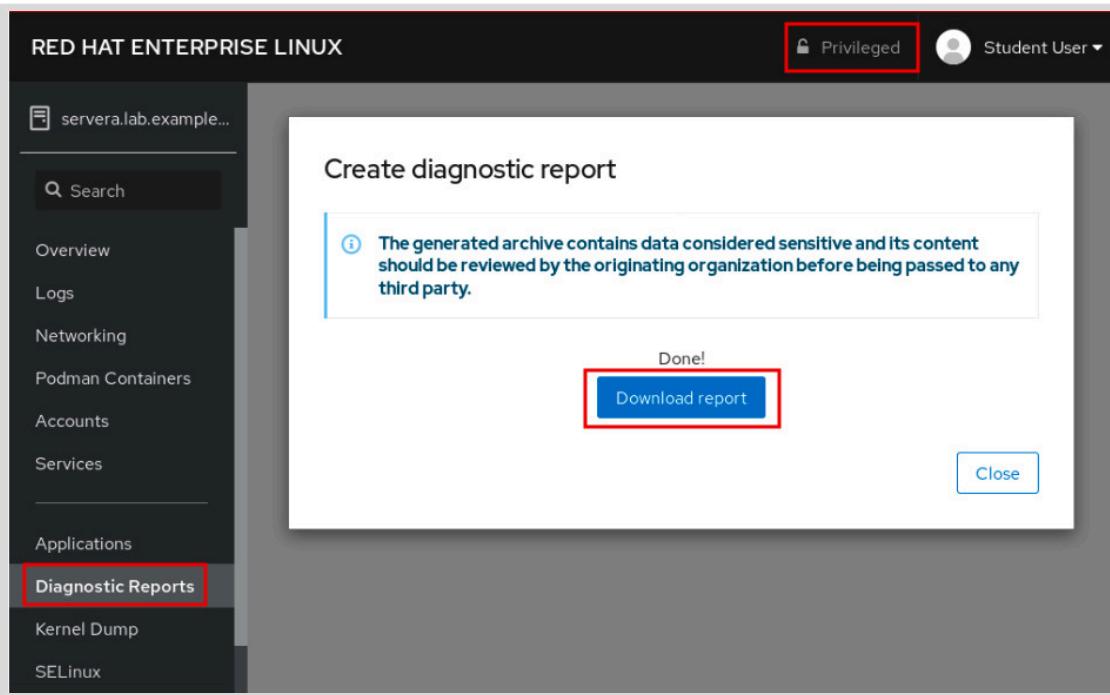


Figure 16.14: Downloading a completed report

Click **Save File** to save the file and complete the process.

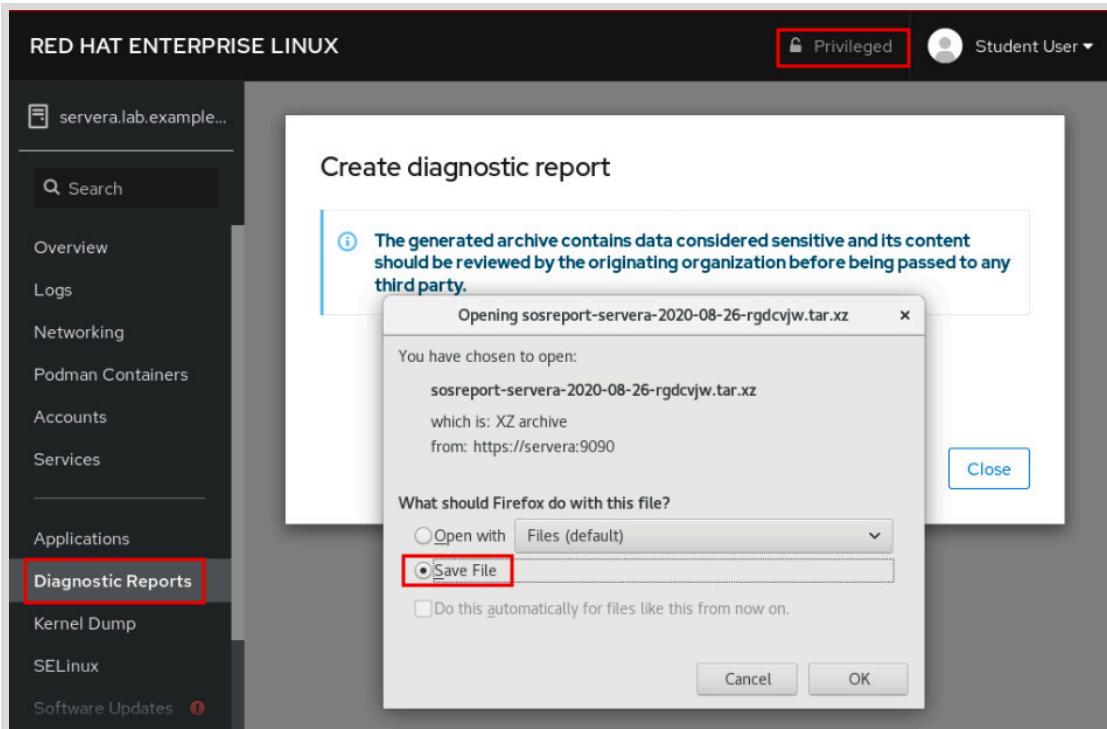


Figure 16.15: Saving a diagnostic report

The completed report is saved to the **Downloads** directory on the system hosting the web browser used to access Web Console. In this example, the host is **workstation**.

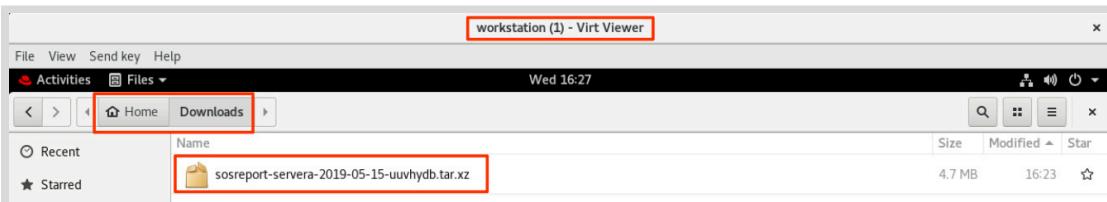


Figure 16.16: Accessing a completed report

Managing System Services with the Web Console

As a privileged user in Web Console, you can stop, start, enable, and restart system services. Additionally, you can configure network interfaces, configure firewall services, administer user accounts, and more. The following images display common examples for using the Web Console's management tools.

System Power Options

Web Console allows you to restart or shut down the system. Log in to Web Console as a privileged user. Click **Overview** on the navigation bar to access system power options.

Select the desired option from the menu on the upper right to either restart or shut down a system.

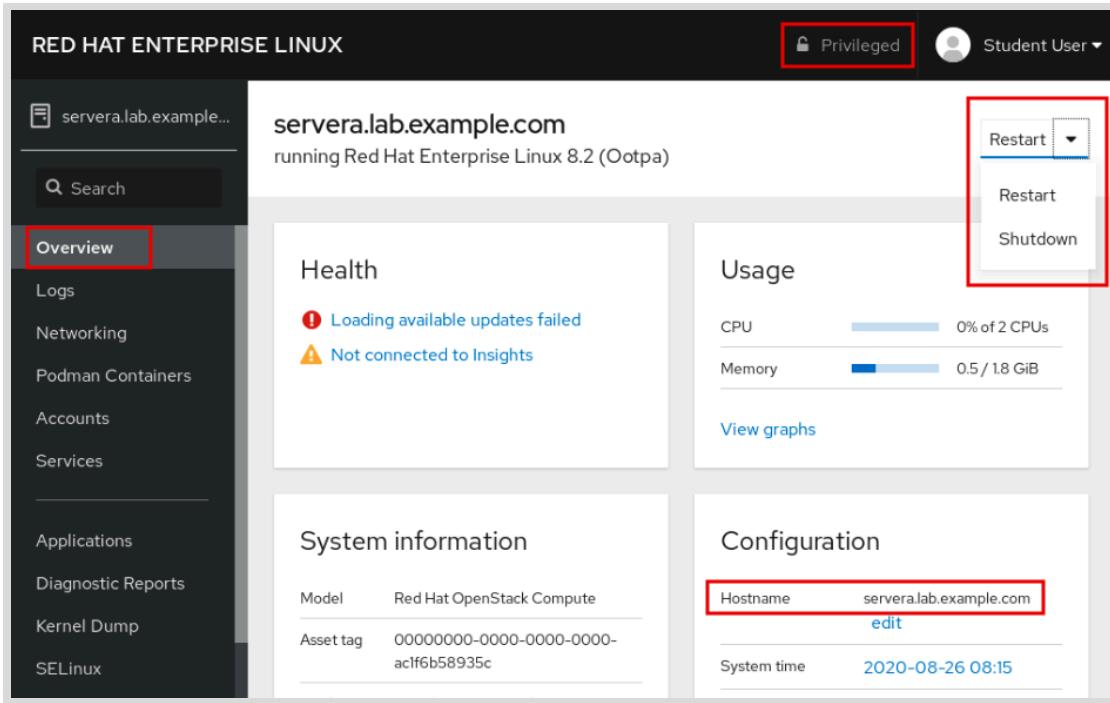


Figure 16.17: System power options

Controlling Running System Services

You can start, enable, disable, and stop services with graphical tools in Web Console. Click **Services** on the navigation bar to access the Web Console's services initial page. To manage services, click **System Services** at the top of the services initial page. Search in the search bar or scroll through the page to select the service you want to manage.

In the example below, select the **chronyd.service** row to open the service management page.

The screenshot shows the 'Services: Initial view' page. The top navigation bar indicates the server is running Red Hat Enterprise Linux 8.2 (Ootpa) and shows a 'Privileged' status and a 'Student User' account. The left sidebar has a 'Services' section highlighted with a red box. The main content area features a 'System Services' tab highlighted with a red box, followed by 'Targets', 'Sockets', 'Timers', and 'Paths'. A search bar and a filter dropdown are present. A table lists various system services with columns for Name, Description, Status, and Action. The 'chronyd' service is highlighted with a red box in the table.

Name	Description	Status	Action
arp-ethers	Load static arp entries	inactive (dead)	Disabled
atd	Job spooling tools	active (running)	Enabled
auditd	Security Auditing Service	active (running)	Enabled
auth-rpcgss-module	Kernel Module supporting RPCSEC_GSS	inactive (dead)	Static
autovt@	autovt@.service Template		
blk-availability	Availability of block devices	inactive (dead)	Disabled
chrony-dnssrv@	chrony-dnssrv@.service Template		
chrony-wait	Wait for chrony to synchronize system clock	inactive (dead)	Disabled
chronyd	NTP client/server	active (running)	Enabled
cloud-config	Apply the settings specified in cloud-config	inactive (dead)	Disabled

Figure 16.18: Services: Initial view

Click **Stop**, **Restart**, or **Disallow running (mask)** as appropriate to manage the service. In this view the service is already running. Additional information related to the service is available by clicking on any of the highlighted links or by scrolling through the service logs displayed below the service management section.

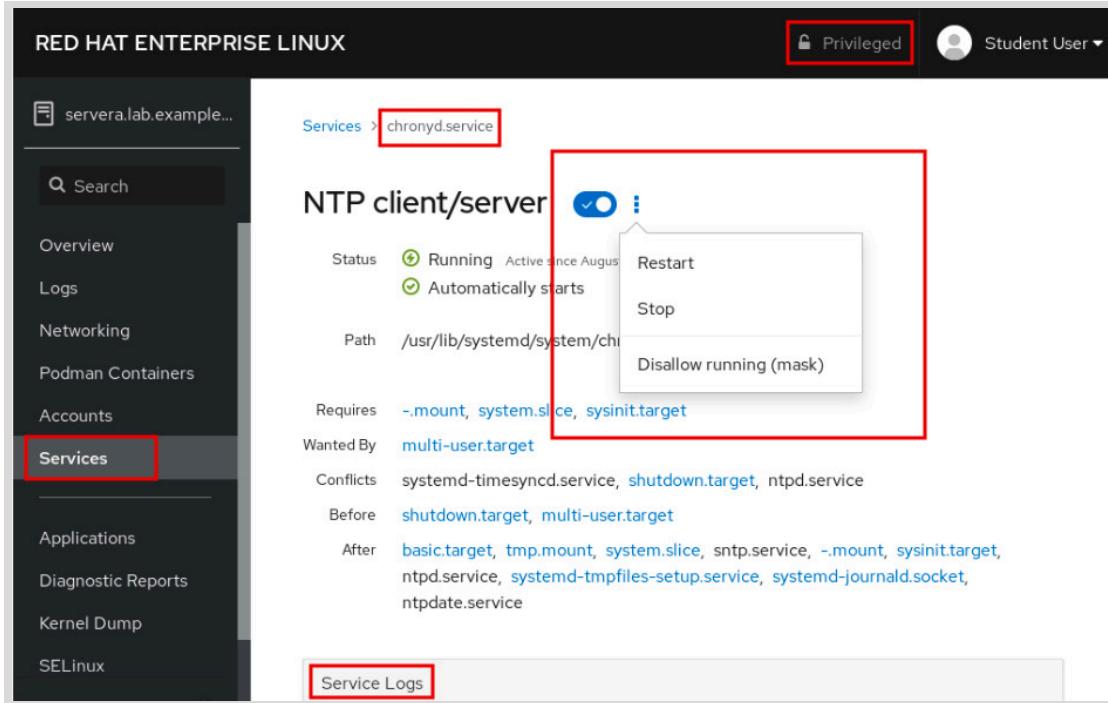


Figure 16.19: Services: Service details and management interface

Configuring Network Interfaces and the Firewall

To manage firewall rules and network interfaces, click **Networking** on the navigation bar. The following example shows how to gather information about network interfaces and how to manage them.

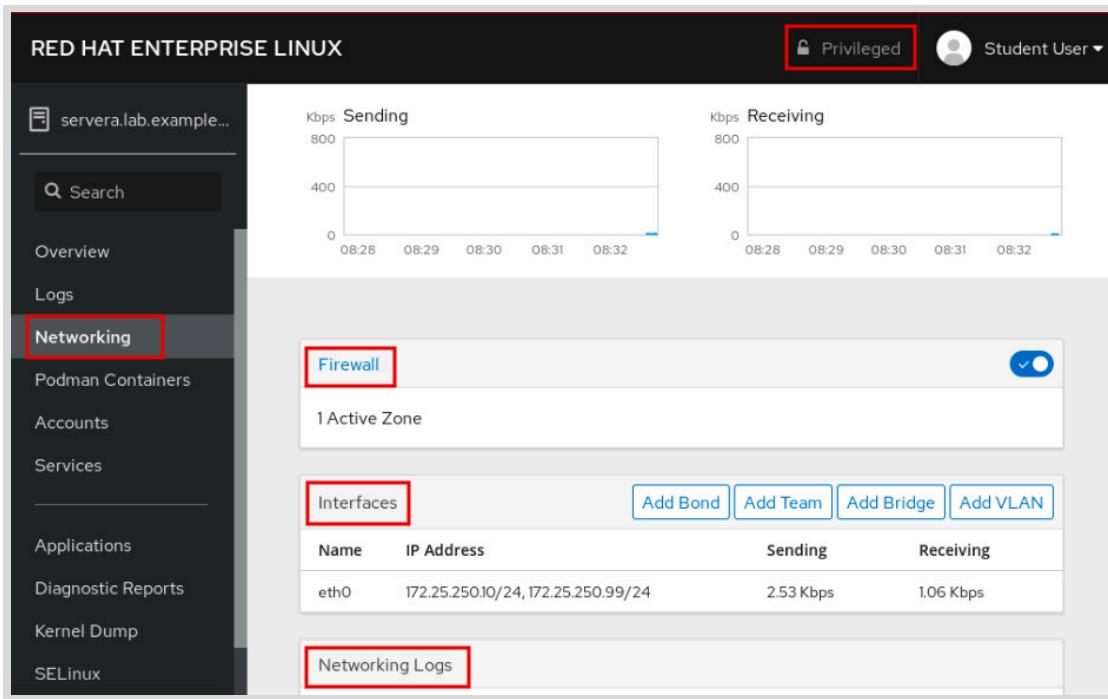


Figure 16.20: Networking: Initial view

Click the desired interface name in the **Interfaces** section to access the management page. In this example, the **eth0** interface is selected.

Interfaces		Add Bond	Add Team	Add Bridge	Add VLAN
Name	IP Address	Sending	Receiving		
eth0	172.25.250.10/24, 172.25.250.100/24	3.84 Kbps	1.58 Kbps		

Figure 16.21: Networking: Interfaces

The top part of the management page displays network traffic activity for the selected device. Scroll down to view configuration settings and management options.

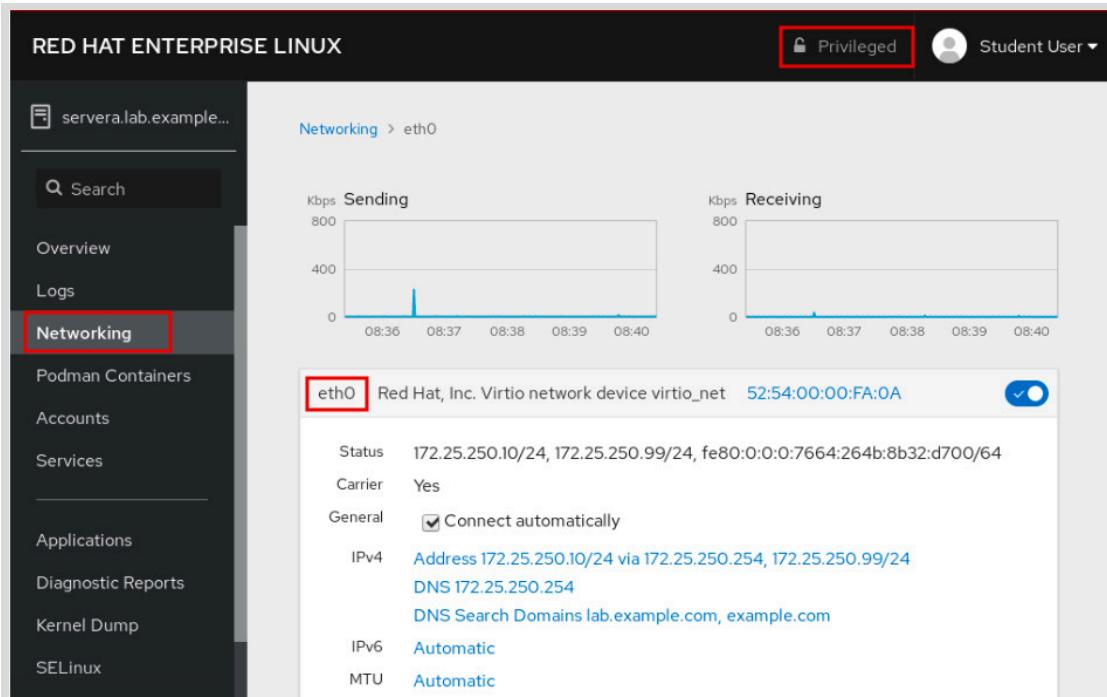


Figure 16.22: Networking: Interface details

To modify or add configuration options to an interface, click the highlighted links for the desired configuration. In this example, the **IPv4** link shows a single IP address and netmask, **172.25.250.10/24** for the **eth0** network interface. To add an additional IP address to the **eth0** network interface, click the highlighted link.



Figure 16.23: Networking: eth0 configuration section

Click **+** on the right side of the **Manual** list selection to add an additional IP address. Enter an IP address and network mask in the appropriate fields. Click **Apply** to activate the new settings.

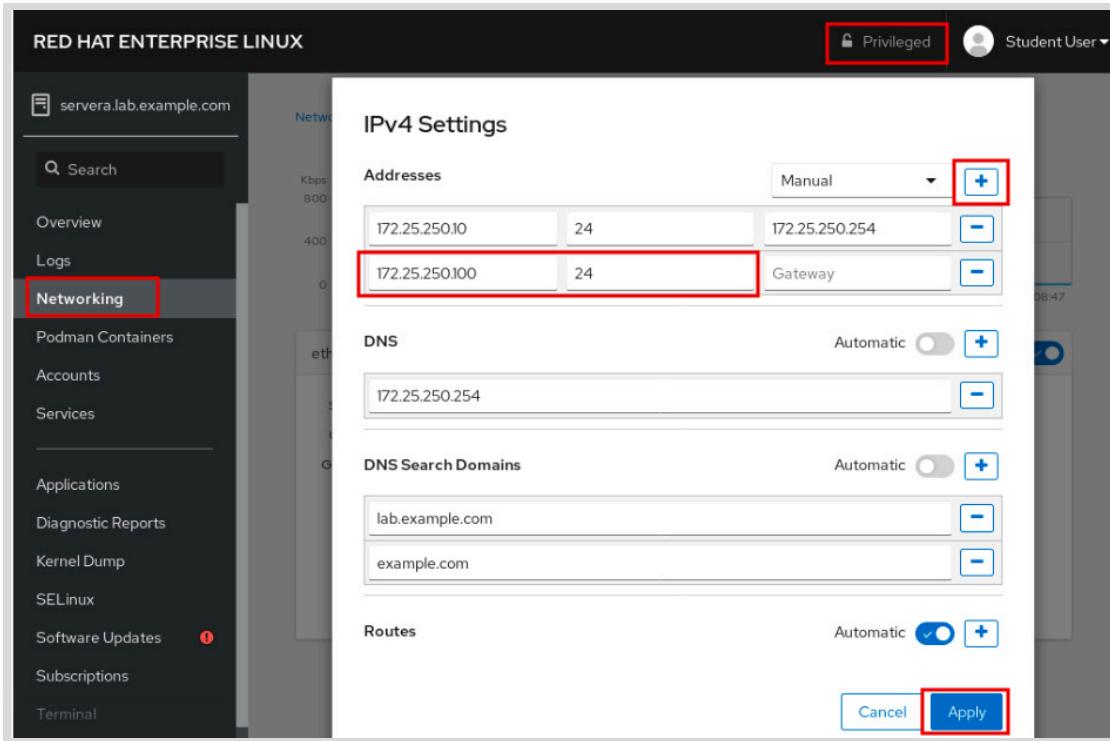


Figure 16.24: Adding an IP address to an existing interface

The display automatically switches back to the interface's management page where you can confirm the new IP address.

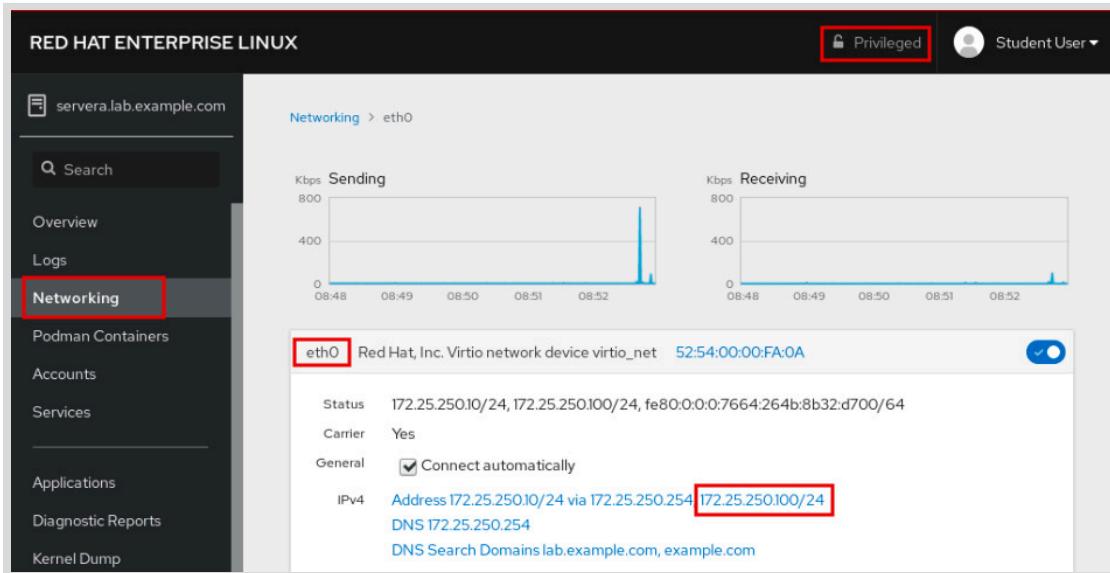


Figure 16.25: Confirming the new IP address

Administering User Accounts

As a privileged user you can create new user accounts in Web Console. Click **Accounts** on the navigation bar to view existing accounts. Click **Create New Account** to open the account management page.

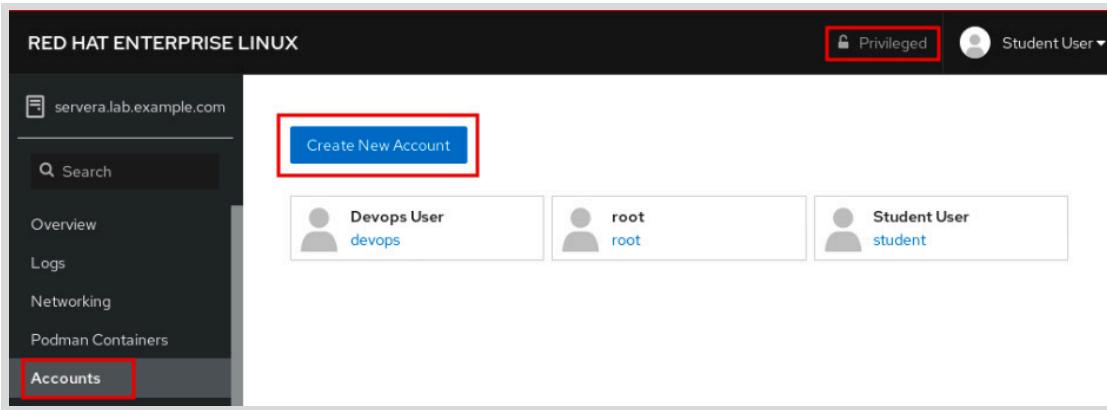


Figure 16.26: Existing user accounts

Enter the information for the new account and then click **Create**.

The screenshot shows the 'Create New Account' dialog box. It contains fields for Full Name ('New User'), User Name ('nuser'), Password ('*****'), Confirm ('*****'), and Access ('Lock Account' checkbox). At the bottom are 'Cancel' and 'Create' buttons, with 'Create' highlighted with a red box.

Figure 16.27: Creating a new account

The display automatically switches back to the account management page where you can confirm the new user account.

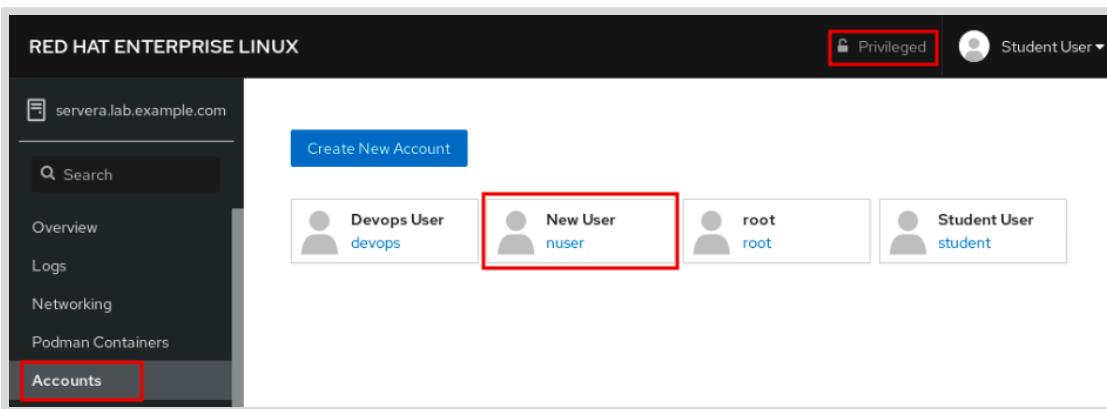


Figure 16.28: Account management page



References

cockpit(1), **cockpit-ws(8)**, and **cockpit.conf(5)** man pages

For more information, refer to *Managing systems using Web Console* in the guide to using Cockpit for managing systems in Red Hat Enterprise Linux 8 at https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/8/html-single/managing_systems_using_the_web_console/

► Guided Exercise

Analyzing and Managing Remote Servers

In this exercise, you will enable and access Web Console on a server to manage it and to diagnose and resolve issues.

Outcomes

You should be able to use Web Console to monitor basic system features, inspect log files, create user accounts, and access the terminal.

Before You Begin

Log in as the **student** user on **workstation** using **student** as the password.

On **workstation**, run the **lab support-cockpit start** command. This command runs a start script to determine and alter whether the hosts **servera** and **serverb** are reachable on the network.

```
[student@workstation ~]$ lab support-cockpit start
```

- ▶ 1. Use the **ssh** command to log in to **servera** as the **student** user. The systems are configured to use SSH keys for authentication, therefore a password is not required to log in to **servera**.

```
[student@workstation ~]$ ssh student@servera
Activate the web console with: systemctl enable --now cockpit.socket
```

```
[student@servera ~]$
```

- ▶ 2. Web Console is already installed on the system, but it is not activated. Enable and start the **cockpit** service.
 - 2.1. Use the **systemctl enable --now cockpit.socket** command to enable the Web Console service. Use the **sudo** command to get superuser privileges, and when prompted use **student** as the password.

```
[student@servera ~]$ sudo systemctl enable --now cockpit.socket
[sudo] password for student: student
Created symlink /etc/systemd/system/sockets.target.wants/cockpit.socket -> /usr/lib/systemd/system/cockpit.socket.
```

- ▶ 3. On **workstation**, open Firefox and log in to the Web Console interface at **servera.lab.example.com**. Log in as **student** using **student** as the password.
 - 3.1. Open Firefox and navigate to **https://servera.lab.example.com:9090**.
 - 3.2. Accept the self-signed certificate by adding it as an exception.

- 3.3. Clear the check box **Reuse my password for privileged tasks**.

- 3.4. Log in as **student** using **student** as the password.

You are now logged in as a normal user, with minimal privileges.

▶ **4.** Verify your current authorization within the Web Console interface.

- 4.1. Click **Terminal** on the left navigation bar to access the terminal.

A terminal session opens with the **student** user already logged in. Use the **id** command to confirm that command execution works in the embedded terminal.

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

- 4.2. Click **Accounts** on the left navigation bar to manage users.

Move the mouse pointer over the **Create New Account** button located in the upper-left corner. Notice that the **student** user is not permitted to create new accounts.

- 4.3. Click the **Student User** link.

On the **student** user's account details page, notice that the user is only permitted to set a new password or to add authorized SSH public keys.

- 4.4. In the upper-right corner, click **Student User → Log Out**.

▶ **5.** Access Web Console with administrative privileges.

- 5.1. Log back in to the Web Console interface as the **student** user with **student** as the password, but this time select the **Reuse my password for privileged tasks** check box.

- 5.2. To verify administrative access, confirm that the **Privileged** label is displayed next to the **Student User** account name at the upper-right side of the Web Console interface.

▶ **6.** To investigate system statistics, click **Overview** on the left navigation bar.

This page displays various basic operating system statistics, such as current load, disk usage, disk I/O, and network traffic.

▶ **7.** To inspect system logs, click **Logs** on the left navigation bar.

This page displays the **systemd** system logs. Use the buttons located in the upper-left corner to modify how log entries are displayed based on date and the severity of the logs.

- 7.1. Click the **Severity** list and choose **Everything**.

- 7.2. Based on the current day of the month, click any log entry from the list. A log entry detail page opens with additional information about the event, such as the host name, the SELinux context, or the PID number of the process that the entry corresponds to.

▶ **8.** Add a second IP address to an existing network interface device.

- 8.1. Click **Networking** on the left navigation bar.

This page displays details of the current network configuration for **servera**, as well as real-time network statistics, firewall configuration, and log entries related to networking.

- 8.2. Scroll down to the the **Interfaces** section and click the row for the network interface name.

A details page displays real-time network statistics, as well as the current configuration for that network interface.

- 8.3. Click the **Address 172.25.250.10/24 via 172.25.250.254** link.

An **IPv4 Settings** window opens where you can change the network interface configuration.

- 8.4. In the **IPv4 Settings** window, click **+** next to **Manual**.

- 8.5. In the **Address** text box, enter **172.25.250.99** as the second IP Address.

- 8.6. In the **Prefix length or Netmask** text box, enter **24** as the netmask value.

- 8.7. Click **Apply** to save the new network configuration.

Notice that the new configuration is immediately applied. The new IP Address is visible in the **IPv4** line.

▶ **9.** Create a new user account.

- 9.1. Click **Accounts** on the left navigation bar.

- 9.2. Click **Create New Account**.

- 9.3. In the **Create New Account** window, add the following details:

Field	Value
Full Name	manager1
User Name	manager1
Password	redh@t!23
Confirm	redh@t!23

- 9.4. Click **Create**.

▶ **10.** Access a terminal session within Web Console to add the **manager1** user to the **wheel** group.

- 10.1. Click **Terminal** on the left navigation bar.

- 10.2. Use the **id manager1** command to view the group membership of the **manager1** user.

```
[student@servera ~]$ id manager1
uid=1001(manager1) gid=1001(manager1) groups=1001(manager1)
[student@servera ~]$
```

- 10.3. Use the **sudo usermod -aG wheel manager1** command to add **manager1** to the **wheel** group.

```
[student@servera ~]$ sudo usermod -aG wheel manager1
[sudo] password for student:
[student@servera ~]$
```

- 10.4. Use the **id manager1** command to verify that **manager1** is a member of the **wheel** group.

```
[student@servera ~]$ id manager1
uid=1001(manager1) gid=1001(manager1) groups=1001(manager1),10(wheel)
[student@servera ~]$
```

► 11. Enable and start the Kernel process accounting service (**psacct**).

- 11.1. Click **Services** on the left navigation bar.
- 11.2. Search for the **Kernel process accounting** service. Click the service link. A details page displays the service status as disabled.
- 11.3. Click the **Start and Enable** button next to the service name.
- 11.4. The service is now enabled and started.

► 12. Log off from the Web Console interface.

► 13. Log off from **servera**.

```
[student@servera ~]$ exit
[student@workstation ~]$
```

Finish

On **workstation**, run the **lab support-cockpit finish** script to finish this exercise.

```
[student@workstation ~]$ lab support-cockpit finish
```

This concludes the guided exercise.

Getting Help From Red Hat Customer Portal

Objectives

After completing this section, you should be able to describe key resources available through the Red Hat Customer Portal and use them to find information from Red Hat documentation and the Knowledgebase.

Accessing Support Resources on the Red Hat Customer Portal

The Red Hat Customer Portal (<https://access.redhat.com>) provides customers access to documentation, downloads, tools, and technical expertise. Customers can search for solutions, FAQs, and articles through the Knowledgebase. From the Customer Portal, you can:

- Access official product documentation.
- Submit and manage support tickets.
- Manage software subscriptions and entitlements.
- Obtain software downloads, updates, and evaluations.
- Consult tools that can help you optimize the configuration of your systems.

Parts of the site are accessible to everyone, and other areas are only available to customers with active subscriptions. Get help accessing the Customer Portal at <https://access.redhat.com/help/>.

Getting Oriented to the Customer Portal

You can access the Red Hat Customer Portal through a web browser. This section introduces the Customer Portal Tour. The tour can be found at <https://access.redhat.com/start>.

The tour is a very useful tool for discovering all the portal has to offer and how to get the most out of your Red Hat subscription. After you have logged in to the Red Hat Customer Portal, click **Tour the Customer Portal**.

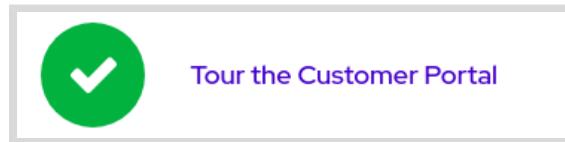


Figure 16.29: Tour the Customer Portal

The **WELCOME TO THE RED HAT CUSTOMER PORTAL** window opens with two options: **CLOSE** and **NEXT**. Click **NEXT** to start the tour. This is the first of a sequence of windows that highlight different parts of the interface.

The Top Navigation Bar

The first three stops on the Customer Portal Tour can be found on the top navigation bar of the Red Hat Customer Portal website:



Figure 16.30: Top Navigation Bar

Subscriptions opens a new page where you can manage your registered systems and your subscriptions and entitlements usage. It lists information about errata that apply and allows you to create *activation keys* that you can use when registering systems to ensure they get entitlements from the correct subscriptions. Note that if you are part of an organization, your Organization Administrator can limit your access to this page.

Downloads opens a new page which gives you access to your product downloads and to request evaluation entitlements for products for which you do not have entitlements.

Support Cases opens a new page which provides access to create, track, and manage your support cases through the Case Management system, assuming that your organization has authorized that level of access.

Your name is the title for the **User Menu**, which allows you to manage your account, accounts for which you are Organization Administrator, your personal profile, and options for email notifications of new content that is available.

The globe icon opens the **Select Your Language** menu to specify your language preferences for Customer Portal.

Topics Menus

Underneath the top navigation bar on the Customer Portal's main page are menus that you can use to navigate to four major categories of resources available on the site.

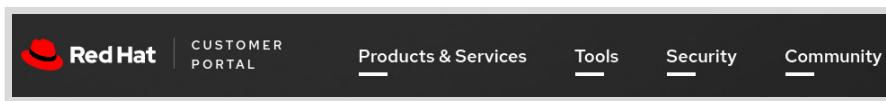
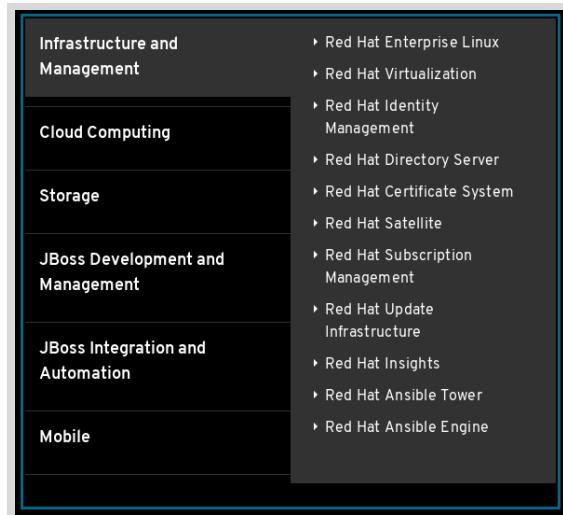
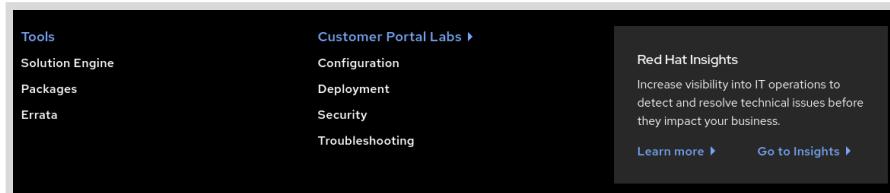


Figure 16.31: Resources Menus

Products & Services provides access to the *Product Hubs*, pages that provide access to product-specific evaluations, overviews, getting started guides, and other product support information. You can also access documentation for Red Hat products, direct links to the Knowledgebase of support articles, and information on support policies and how to contact Red Hat Support.

**Figure 16.32: Products and Services**

The **Tools** menu provides links to tools to help you succeed with Red Hat products. The Solution Engine section provides you with an efficient way to search for solutions to your problems quickly, by product, and opening a support ticket if you do not find a satisfactory solution. The Customer Portal Labs section provides a collection of web-based applications and tools to help you improve performance, diagnose issues, identify security problems, and optimize your configurations. For example, the Product Life Cycle Checker allows you to select a particular product and view its support life cycle schedule. Another tool, the Rescue Mode Assistant, helps you reset the root password of a system, generate diagnostic reports, or fix boot-time problems with file systems. But there are many other tools available at that site.

**Figure 16.33: Tools menu in Customer Portal**

The Security section provides access to the *Red Hat Product Security Center* at <https://access.redhat.com/security/>. This section also provides information about high-profile security issues, access to the Red Hat CVE Database, the Security channel of the Red Hat Blog, and resources about Red Hat's security response process and how we rate issues and resolve them.

Finally, the Community section is a place where Red Hat experts, customers, and partners can communicate and collaborate. Discussion forums, blogs, and information about upcoming events in your area are available here.

**Note**

You should complete the entire tour at *Getting Started with Red Hat* [<https://access.redhat.com/start>], including the sections on how to personalize your Customer Portal experience and exploring the benefits of your Red Hat subscription, to get the full story about the Customer Portal. You will need at least one active subscription on your Customer Portal account to access this page.

Searching the Knowledgebase with the Red Hat Support Tool

The Red Hat Support Tool utility, **redhat-support-tool**, provides a text-based interface that allows you to search Knowledgebase articles and to file support cases on the Customer Portal from your system's command line. The tool does not have a graphical interface and, because it interacts with the Red Hat Customer Portal, it requires internet access. Run the **redhat-support-tool** command using any terminal or SSH connection.

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

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

When first invoked, **redhat-support-tool** prompts for Red Hat Customer Portal subscriber login information. To avoid repetitively supplying this information, the tool asks to store account information in the user's home directory (`~/.redhat-support-tool/redhat-support-tool.conf`). If issues are all filed through a particular Red Hat Customer Portal account, the **--global** option can save account information to `/etc/redhat-support-tool.conf`, along with other system-wide configuration. The tool's **config** command modifies tool configuration settings.

The **redhat-support-tool** command allows subscribers to search and display Knowledgebase content from the Red Hat Customer Portal. The Knowledgebase permits keyword searches, similar to the **man** command. You 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:

```
[user@host ~]$ 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 the Red Hat Customer
Portal using Red Hat Subscription-Manager
2 [ 265523:VER] Enabling or disabling a repository using Red Hat Subscription
Management
```

Chapter 16 | Analyzing Servers and Getting Support

```
3 [ 100423:VER] Why does subscription-manager list return: "No Installed  
Products found" ?  
...output omitted...  
Select a Solution: 1
```

Select article number 1 as above and you are prompted to select the section of the document to read. Finally, use the **Q** key to quit the section you are in, or use it repeatedly to quit the **redhat-support-tool** command.

```
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 the Red Hat Customer Portal using Red  
Hat Subscription-Manager  
URL: https://access.redhat.com/solutions/253273  
Created On: None  
Modified On: 2017-11-29T15:33:51Z  
  
(END) q  
Section:  
Section: q  
  
Select a Solution: q  
  
Command (? for help): q  
[user@hosts ~]#
```

Accessing Knowledgebase Articles by Document ID

Locate online articles directly using the tool's **kb** command with the Knowledgebase document ID. A returned document scrolls on the screen without pagination, but you can redirect it to a file to save it and use **less** to scroll through it a screen at a time.

```
[user@host ~]$ redhat-support-tool kb 253273  
  
Title  
=====  
How to register and subscribe a system to the Red Hat Customer Portal using Red  
Hat Subscription-Manager  
URL: https://access.redhat.com/solutions/253273  
Created On: None  
Modified On: 2017-11-29T15:33:51Z
```

```

Issue
=====
* How to register a new `Red Hat Enterprise Linux` system to the Customer Portal
  using `Red Hat Subscription-Manager`
...output omitted...

```

Managing Support Cases with Red Hat Support Tool

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

Preparing a Bug report

Before contacting Red Hat Support, it is important to 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 that 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. This includes loss of production data or malfunctioning production systems. The situation halts your business operations and no procedural workaround exists.
<i>High</i> (Severity 2)	A problem where the software is functioning but use in a production environment is severely reduced. The situation is causing a high impact to 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. Business continues to function using a procedural workaround. For development environments, the situation is causing problems migrating your project into production.
<i>Low</i> (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 using a procedural workaround.

Managing a Bug Report with redhat-support-tool

You can create, view, modify, and close Red Hat Support cases using **redhat-support-tool**. When support cases are in an **opened** or **maintained** status, users may attach files or documentation, such as diagnostic reports (sosreport). The tool uploads and attaches files to cases.

Case details including the product name, 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, a new case is opened. The **--product** and **--version** options are specified.

```
[user@host ~]$ 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 Urgent
2 High
3 Normal
4 Low
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.
```

If the **--product** and **--version** options are not specified the **redhat-support-tool** provides a list of choices for those options.

```
[user@host ~]$ redhat-support-tool
Welcome to the Red Hat Support Tool.
Command (? for help): opencase
Do you want to use the default product - "Red Hat Enterprise Linux" (y/N)?: y
...output omitted...
29 7.4
30 7.5
31 7.6
32 8.0 Beta
Please select a version (or 'q' to exit): 32
Please enter a summary (or 'q' to exit): yum fails to install apache
Please enter a description (Ctrl-D on an empty line when complete):
yum cannot find correct repo
1 Urgent
2 High
3 Normal
4 Low
Please select a severity (or 'q' to exit): 4
Would you like to use the default (Ungrouped Case) Case Group (y/N)? : y
```

```
Would you like to see if there's a solution to this problem before opening a
support case? (y/N) N
```

```
-----
```

```
Support case 010355678 has successfully been opened.
```

Attaching Diagnostic Information to a Support Case

Including diagnostic information can lead to a quicker resolution. Attach the `sosreport` when the case is opened. 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 does not exist, an administrator can generate and attach one later. Use the `redhat-support-tool addattachment` command to attach the report.

Support cases can also be viewed, modified, and closed by 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
```

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

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

backtrace is a report of the active stack frames at the point of a crash dump and provides onsite diagnostics. One of the options of the **redhat-support-tool** is to 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, and oVirt, can be parsed to recognize problem symptoms. The log files can 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.

Joining Red Hat Developer

One other useful resource available from Red Hat is Red Hat Developer. Hosted at <https://developer.redhat.com>, this program provides subscription entitlements to Red Hat software for development purposes, documentation, and premium books from our experts on microservices, serverless computing, Kubernetes, and Linux. A blog, links to information about upcoming events and training, and other help resources are also available, as well as links to Red Hat Customer Portal.

Registration is free, and can be completed at <https://developer.redhat.com/register>.



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/>

► Guided Exercise

Getting Help from Red Hat Customer Portal

In this exercise, you will generate a diagnostics report using Web Console.

Outcomes

You should be able to generate a diagnostics report using Web Console which could be submitted to Red Hat Customer Portal as part of a support case.

Before You Begin

Log in as the **student** user on **workstation** using **student** as the password.

From **workstation**, run the **lab support-portal start** command. The command runs a start script that determines if **servera** is reachable on the network. It also starts and enables Web Console on **servera**.

```
[student@workstation ~]$ lab support-portal start
```

- 1. From **workstation** use the **ssh** command to log into **servera** as the **student** user.

```
[student@workstation ~]$ ssh student@servera
Web console: https://servera.lab.example.com:9090/ or https://172.25.250.10:9090/
[student@servera ~]$
```

- 2. Use the **systemctl** command to confirm that the **cockpit** service is running. Enter **student** as the password when prompted.

```
[student@servera ~]$ systemctl status cockpit.socket
● cockpit.socket - Cockpit Web Service Socket
  Loaded: loaded (/usr/lib/systemd/system/cockpit.socket; enabled; vendor preset: disabled)
  Active: active (listening) since Thu 2019-05-16 10:32:33 IST; 4min 37s ago
    Docs: man:cockpit-ws(8)
   Listen: [::]:9090 (Stream)
  Process: 676 ExecStartPost=/bin/ln -snf active.motd /run/cockpit/motd
             (code=exited, status=0/SUCCESS)
  Process: 668 ExecStartPost=/usr/share/cockpit/motd/update-motd localhost
             (code=exited, status=0/SUCCESS)
    Tasks: 0 (limit: 11405)
   Memory: 1.5M
      CGroup: /system.slice/cockpit.socket
              ...output omitted...
```

- 3. Log out from **servera**.

```
[student@servera ~]$ exit  
[student@workstation ~]$
```

- ▶ 4. On **workstation**, open *Firefox* and log in to the Web Console interface running on **servera.lab.example.com** as the **root** user with **redhat** as the password.
 - 4.1. Open *Firefox* and go to the **https://servera.lab.example.com:9090** address.
 - 4.2. If prompted, accept the self-signed certificate by adding it as an exception.
 - 4.3. Log in as the **root** user with **redhat** as the password. You are now logged in as a privileged user, which is necessary to create a diagnostic report.
 - 4.4. Click **Diagnostic Reports** in the left navigation bar. Click on **Create Report**. The report takes a few minutes to create.
- ▶ 5. When the report is ready, click on **Download report**. Save the file.
 - 5.1. Click the **Download report** button, followed by the **Save File** button.
 - 5.2. Click the **Close** button.
 - 5.3. Log out from the Web Console interface.

Finish

On **workstation**, run the **lab support-portal finish** script to complete this exercise.

```
[student@workstation ~]$ lab support-portal finish
```

This concludes the guided exercise.

Detecting and Resolving Issues with Red Hat Insights

Objectives

After completing this section, you should be able to use Red Hat Insights to analyze servers for issues, remediate or resolve them, and confirm the solution worked.

Introducing Red Hat Insights

Red Hat Insights is a predictive analytics tool to help you identify and remediate threats to security, performance, availability, and stability on systems in your infrastructure running Red Hat products. Insights is delivered as a Software-as-a-Service (SaaS) product, so that you can deploy and scale it quickly with no additional infrastructure requirements. In addition, you can immediately take advantage of the latest recommendations and updates from Red Hat specific to your deployed systems.

Red Hat regularly updates the knowledge base used by Insights, based on common support risks, security vulnerabilities, known-bad configurations, and other issues identified by Red Hat. Actions to mitigate or remediate these issues are validated and verified by Red Hat. This support allows you to proactively identify, prioritize, and resolve issues before they become a larger problem.

For each detected issue, Insights provides estimates of the risk presented and recommendations on how to mitigate or remediate the problem. These recommendations may provide materials such as Ansible Playbooks or human-readable step-by-step instructions to help you resolve the issue.

Insights tailors recommendations to each system registered to the service. You install each client system with an agent that collects metadata about the runtime configuration of the system. This data is a subset of what you might provide to Red Hat Support using the **sosreport** command in order to resolve a support ticket. You can limit or obfuscate the data that your clients send. This blocks some of the analytic rules from operating, depending on what you limit.

Almost immediately after you register a server and it completes the initial system metadata synchronization, you should be able to see your server and any recommendations for it in the Insights console in Red Hat Cloud Portal.

Insights currently provides predictive analytics and recommendations for these Red Hat products:

- Red Hat Enterprise Linux 6.4 and later
- Red Hat Virtualization 4 and later
- Red Hat OpenShift Container Platform
- Red Hat OpenStack Platform 7 and later

Describing the Insights Architecture

When you register a system with Insights, it immediately sends metadata about its current configuration to the Insights platform. After registration, the system periodically updates the metadata provided to Insights. The system sends the metadata using TLS encryption to protect it in transit.

When Insights receives the data, it analyses it and displays the result on the Insights web console at <https://cloud.redhat.com/insights>.

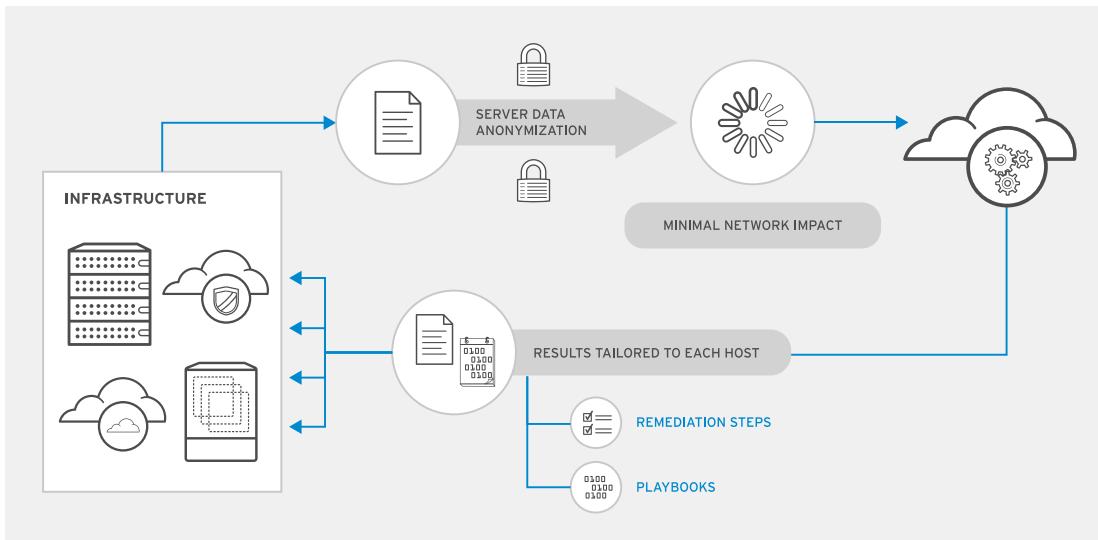


Figure 16.34: Insights high-level architecture

Installing Insights Clients

Insights is included with Red Hat Enterprise Linux 8 as part of the subscription. Earlier versions of Red Hat Enterprise Linux servers require installing the *insights-client* package on the system.



Important

The *insights-client* package replaces the *redhat-access-insights* package starting with Red Hat Enterprise Linux 7.5.

If your system is registered for software entitlements through the Customer Portal Subscription Management service, you can activate Insights with one command. Use the **`insights-client --register`** command to register the system.

```
[root@host ~]# insights-client --register
```

The Insights client periodically updates the metadata provided to Insights. Use the **`insights-client`** command to refresh the client's metadata at any time.

```
[root@host ~]# insights-client
Starting to collect Insights data for host.example.com
Uploading Insights data.
Successfully uploaded report for host.example.com.
View details about this system on cloud.redhat.com:
https://cloud.redhat.com/insights/inventory/dc480efd-4782-417e-a496-cb33e23642f0
```

Registering a RHEL System with Insights

To register a RHEL server to Insights, the process is as follows:

- Interactively register the system with the Red Hat Subscription Management service.

```
[root@host ~]# subscription-manager register --auto-attach
```

- Make sure that the `insights-client` package is installed on the system. In RHEL 7, this package is in the **rhel-7-server-rpms** channel.

**Note**

This step is not required on Red Hat Enterprise Linux 8 systems.

```
[root@host ~]# yum install insights-client
```

- Use the `insights-client --register` command to register the system with the Insights service and upload initial system metadata.

```
[root@host ~]# insights-client --register
```

- Confirm that the system is visible under **Inventory** in the Insights web console at <https://cloud.redhat.com/insights>.

Name	Tags	Last seen
host.example.com	0	32 minutes ago
utility.example.com	0	40 minutes ago
rhel8.local	0	8 hours ago
rhel8.phoenix.home.lan	0	8 hours ago

Figure 16.35: Insights Inventory on the Cloud Portal

Navigating the Insights Console

Insights provides a family of services that you access through the web console at <https://cloud.redhat.com/insights>.

Detecting Configuration Issues Using the Advisor Service

The Advisor service reports configuration issues that impact your systems. You access the service from the **Advisor → Recommendations** menu.

The screenshot shows the Red Hat Advisor interface under the 'Advisor recommendations' tab. It displays three system issues:

- Issue 1:** Incident: Decreased stability and/or performance due to filesystem over 95% capacity and available space is less than 100MB. Added 5 months ago. Total risk: Important (orange). Risk of change: Low (yellow). System count: 1. Ansible: No.
- Issue 2:** Traffic occurs or services are allowed unexpectedly when firewall zone drifting is enabled. Added 5 months ago. Total risk: Moderate (yellow). Risk of change: Moderate (yellow). System count: 2. Ansible: No.
- Issue 3:** Decreased security: Yum GPG verification disabled (third-party repos). Added 3 years ago. Total risk: Low (blue). Risk of change: Very Low (light blue). System count: 1. Ansible: No.

Figure 16.36: Recommendations from the Advisor Service

For each issue, Insights provides additional information to help you understand the problem, prioritize work to address it, determine what mitigation or remediation is available, and automate resolution with an Ansible Playbook. Insights also provides links to Knowledgebase articles on the Customer Portal.

The screenshot shows the detailed view for the first issue from Figure 16.36:

Description: File systems nearing full capacity or inode usage can cause performance issues because blocks must be used from different block groups. Besides, file systems at or exceeding capacity will have stability issues because applications will no longer be able to write to the file system.

Total risk: Important (orange). The total risk of this remediation is **important**, based on the combination of likelihood and impact to remediate. Critical likelihood (red bar) and Medium impact (yellow bar).

Risk of change: Low (yellow). The risk of change is **low**, because the change does not require that a system be taken offline. System reboot is **not required**.

Figure 16.37: Details of an Issue

The Advisor service evaluates the risk that an issue presents to your system in two categories.

Total risk

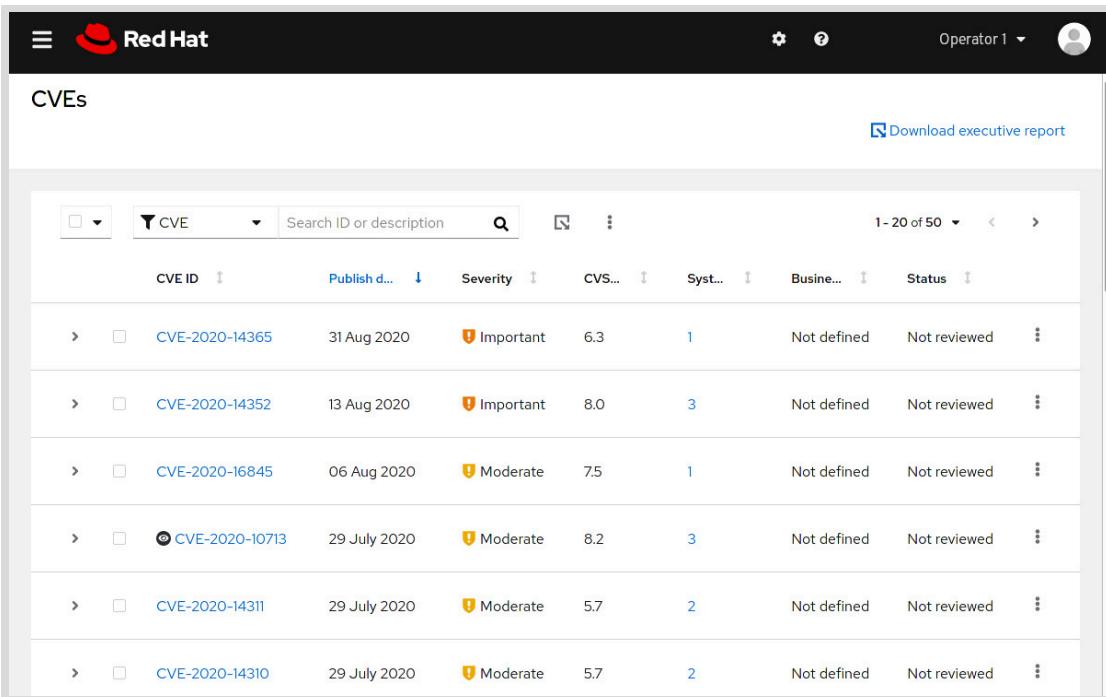
Indicates the impact of the issue on your system.

Risk of change

Indicates the impact of the remediation action to your system. For example, you may have to restart the system.

Assessing Security Using the Vulnerability Service

The Vulnerability service reports *Common Vulnerabilities and Exposures (CVEs)* that impact your systems. You access the service from the **Vulnerability** → **CVEs** menu.



The screenshot shows a web-based application interface for managing vulnerabilities. At the top, there's a navigation bar with the Red Hat logo and a user profile for 'Operator 1'. Below the header, the page title is 'CVEs'. On the right side of the title, there's a link to 'Download executive report'. The main content area is a table listing six CVE entries. The columns in the table are: CVE ID, Publish d..., Severity, CVS..., Syst..., Busine..., and Status. Each row contains a checkbox, a CVE ID link, a publish date, a severity icon (orange exclamation mark), a CVSS score, the number of systems affected, the business unit (Not defined), and the status (Not reviewed). There are also three-dot ellipsis icons at the end of each row.

CVE ID	Publish d...	Severity	CVS...	Syst...	Busine...	Status
CVE-2020-14365	31 Aug 2020	Important	6.3	1	Not defined	Not reviewed
CVE-2020-14352	13 Aug 2020	Important	8.0	3	Not defined	Not reviewed
CVE-2020-16845	06 Aug 2020	Moderate	7.5	1	Not defined	Not reviewed
CVE-2020-10713	29 July 2020	Moderate	8.2	3	Not defined	Not reviewed
CVE-2020-14311	29 July 2020	Moderate	5.7	2	Not defined	Not reviewed
CVE-2020-14310	29 July 2020	Moderate	5.7	2	Not defined	Not reviewed

Figure 16.38: Report from the Vulnerability Service

For each CVE, Insights provides additional information and lists the exposed systems. You can click **Remediate** to create an Ansible Playbook for remediation.

The screenshot shows the Red Hat Insights web interface. On the left, a sidebar menu includes 'Red Hat Insights' (selected), 'Dashboard', 'Advisor', 'Vulnerability' (selected), 'CVEs' (selected), 'Systems', 'Compliance' (selected), 'Patch' (selected), 'Drift', 'Policies', 'Inventory', and 'Remediations'. The main content area displays a detailed view of a vulnerability: CVE-2020-14352. It shows the 'Business risk' as 'Not defined' and 'Status' as 'Not reviewed'. A summary states: 'A flaw was found in librepo. A directory traversal vulnerability was found where it failed to sanitize paths in remote repository metadata. An attacker controlling a remote repository may be able to copy files outside of the destination directory on the targeted system via path traversal. This flaw could potentially result in system compromise via the overwriting of critical system files. The highest threat from this flaw is to users that make use of untrusted third-party repositories.' Publish date is 13 Aug 2020. A link to 'View in Red Hat CVE database' is provided. To the right, a box indicates 'Important severity' (orange shield icon) and a 'CVSS 3.0 base score' of 8.0. Below this, the 'CVSS 3.0 vector' is listed as CVSS:3.1/AV:N/AC:L/PR:L/UI:R/S:U/C:H/I:H/A:H. A section titled 'Exposed systems' lists one host: 'host.example.com' (Status: Not reviewed, Last seen: 1 hour ago). A 'Remediate' button is available for this host.

Figure 16.39: Details of a CVE

Analyzing Compliance Using the Compliance Service

The Compliance service analyses your systems and reports their compliance level to an OpenSCAP policy. The OpenSCAP project implements tools to check the compliance of a system against a set of rules. Insights provides the rules to evaluate your systems against different policies, such as the *Payment Card Industry Data Security Standard (PCI DSS)*.

Updating Packages Using the Patch Service

The Patch service lists the Red Hat Product Advisories applicable to your systems. It can also generate an Ansible Playbook that you can run to update the RPM packages associated with the applicable advisories. To access the list of advisories for a specific system, use the **Patch** → **Systems** menu. Click **Apply all applicable advisories** for a system to generate the Ansible Playbook.

Name	Applicable advisories	Last seen
host.example.com	24	3 hours ago
utility.example.com	25	4 hours ago
rhel8.local	No applicable advisories	11 hours ago
rhel8-laptop	15	11 hours ago

Figure 16.40: Patching a System

Comparing Systems Using the Drift Service

Using the Drift service, you can compare systems, or a system history. This service can help you troubleshoot a system by comparing that system to a similar system, or to a previous system state. You access the service from the **Drift → Comparison** menu.

Fact	State	host.example.com 18 Sep 2020, 11:51 UTC	host.example.com 18 Sep 2020, 07:31 UTC
yum_repos	N/A	N/A	N/A
sap_system	N/A	N/A	N/A
running_processes	N/A	N/A	N/A
last_boot_time	2020-09-18T11:50:06	2020-09-18T07:29:44	
installed_packages	N/A	N/A	N/A
cloud_provider	N/A	N/A	N/A
system_memory	1.78 GB	1.78 GB	
satellite_managed	False	False	
os_release	8.2	8.2	

Figure 16.41: Comparing a System History

In the preceding screen capture, Insights compares the same system at two different times.

Triggering Alerts Using the Policies Service

Using the Policies service, you create rules to monitor your systems and send alerts when a system does not comply with your rules. Insights evaluates the rules every time a system synchronizes its metadata. You access the service from the **Policies** menu.

The screenshot shows the Red Hat Insights web interface. The left sidebar has a dark theme with white text and includes links for Red Hat Insights, Dashboard, Advisor, Vulnerability, Compliance, Patch, Drift, Policies (which is selected and highlighted in blue), Inventory, Remediations, and Register Systems. The main content area is titled 'Policies'. At the top, there's a search bar with 'Name' and 'Filter by name' fields, a 'Create policy' button, and a dropdown showing '1-1 of 1'. Below this is a table with three columns: 'Name', 'Trigger actions', and 'Last triggered'. A single row is listed: 'No GCC' with a mail icon under 'Trigger actions' and a 'Never' status under 'Last triggered'. Below the table, a 'Description' section states: 'Send an alert email when the gcc package is installed.' and 'Last updated 18 Sep 2020 | Created 18 Sep 2020'. At the bottom of the table area, there are two sections: 'Conditions' (with the condition 'facts.installed_packages contains ['gcc']') and 'Trigger actions' (with the action 'Send Email'). Navigation buttons at the bottom of the table area include '1-1 of 1', arrows, and page numbers.

Figure 16.42: Details of a Custom Rule

Accessing the Inventory and the Remediation Playbooks, and Monitoring Subscriptions

The **Inventory** page provides a list of the systems you have registered with Red Hat Insights. The **Last seen** column displays the time of the most recent metadata update for each system. By clicking a system name, you can review its details and directly access the Advisor, Vulnerability, Compliance, and Patch services for that system.

The **Remediations** page lists all the Ansible Playbooks that you created for remediation. You can download the playbooks from that page.

Using the **Subscription Watch** page, you can monitor your Red Hat subscription usage.



References

insights-client(8) and **insights-client.conf(5)** man pages

For more information about Red Hat Insights, refer to the *Product Documentation for Red Hat Insights* at

https://access.redhat.com/documentation/en-us/red_hat_insights

For more information on excluding data collected by Insights, refer to the *Red Hat Insights client data obfuscation* and *Red Hat Insights client data redaction* chapters in the *Client Configuration Guide for Red Hat Insights* at

https://access.redhat.com/documentation/en-us/red_hat_insights/2020-04/html-single/client_configuration_guide_for_red_hat_insights/index

Information on the data collected by Red Hat Insights is available at

System Information Collected by Red Hat Insights

<https://access.redhat.com/articles/1598863>

► Quiz

Detecting and Resolving Issues with Red Hat Insights

Choose the correct answers to the following questions:

► 1. In what order do the following events occur when managing a Red Hat Enterprise Linux system using Red Hat Insights?

1. Red Hat Insights analyzes system metadata to determine which issues and recommendations apply.
 2. The Insights client uploads system metadata to the Red Hat Insights service.
 3. The administrator views the recommended actions in the Red Hat Insights customer portal.
 4. The Insights client collects system metadata on the Red Hat Enterprise Linux system.
- a. 1, 2, 3, 4
b. 4, 2, 1, 3
c. 4, 2, 3, 1
d. 4, 1, 2, 3

► 2. Which command is used to register a client to Red Hat Insights?

- a. `insights-client --register`
- b. `insights-client --no-upload`
- c. `subscription-manager register`
- d. `insights-client --unregister`

► 3. From which page in the Red Hat Insights console can you generate an Ansible Playbook to update the RPM packages on a system?

- a. **Advisor → Recommendations**
- b. **Vulnerability → Systems**
- c. **Patch → Systems**
- d. **Remediations**

► Solution

Detecting and Resolving Issues with Red Hat Insights

Choose the correct answers to the following questions:

► 1. In what order do the following events occur when managing a Red Hat Enterprise Linux system using Red Hat Insights?

1. Red Hat Insights analyzes system metadata to determine which issues and recommendations apply.
 2. The Insights client uploads system metadata to the Red Hat Insights service.
 3. The administrator views the recommended actions in the Red Hat Insights customer portal.
 4. The Insights client collects system metadata on the Red Hat Enterprise Linux system.
- a. 1, 2, 3, 4
 - b. 4, 2, 1, 3
 - c. 4, 2, 3, 1
 - d. 4, 1, 2, 3

► 2. Which command is used to register a client to Red Hat Insights?

- a. `insights-client --register`
- b. `insights-client --no-upload`
- c. `subscription-manager register`
- d. `insights-client --unregister`

► 3. From which page in the Red Hat Insights console can you generate an Ansible Playbook to update the RPM packages on a system?

- a. **Advisor** → **Recommendations**
- b. **Vulnerability** → **Systems**
- c. **Patch** → **Systems**
- d. **Remediations**

Summary

In this chapter, you learned:

- Web Console is a web-based management interface to your server based on the open source Cockpit service.
- Web Console provides graphs of system performance, graphical tools to manage system configuration and inspect logs, and an interactive terminal interfaces.
- Red Hat Customer Portal provides you with access to documentation, downloads, optimization tools, support case management, and subscription and entitlement management for your Red Hat products.
- **redhat-support-tool** is a command-line tool to query Knowledgebase and work with support cases from the server's command line.
- Red Hat Insights is a SaaS-based predictive analytics tool to help you identify and remediate threats to your systems' security, performance, availability, and stability.

Comprehensive Review

Goal

Review tasks from *Red Hat System Administration I*

Objectives

- Review tasks from *Red Hat System Administration I*

Sections

- Comprehensive Review

Lab

- Lab: Managing Files from the Command Line
- Lab: Managing Users and Groups, Permissions and Processes
- Lab: Configuring and Managing a Server
- Lab: Managing Networks
- Lab: Mounting File Systems and Finding Files

Comprehensive Review

Objectives

After completing this section, students should have reviewed and refreshed the knowledge and skills learned in *Red Hat System Administration I*.

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.

Students can refer to earlier sections in the textbook for extra study.

Chapter 1, Getting Started with Red Hat Enterprise Linux

Describe and define open source, Linux, Linux distributions, and Red Hat Enterprise Linux.

- Define and explain the purpose of Linux, open source, Linux distributions, and Red Hat Enterprise Linux.

Chapter 2, Accessing the Command Line

Log in to a Linux system and run simple commands using the shell.

- Log in to a Linux system on a local text console and run simple commands using the shell.
- Log in to a Linux system using the GNOME 3 desktop environment and run commands from a shell prompt in a terminal program.
- Save time by using tab completion, command history, and command editing shortcuts to run commands in the Bash shell.

Chapter 3, Managing Files From the Command Line

Copy, move, create, delete, and organize files while working from the Bash shell.

- Describe how Linux organizes files, and the purposes of various directories in the file-system hierarchy.
- Specify the location of files relative to the current working directory and by absolute location, determine and change your working directory, and list the contents of directories.
- Create, copy, move, and remove files and directories.
- Make multiple file names reference the same file using hard links and symbolic (or "soft") links.
- Efficiently run commands affecting many files by using pattern matching features of the Bash shell.

Chapter 4, Getting Help in Red Hat Enterprise Linux

Resolve problems by using local help systems.

- Find information in local Linux system manual pages.
- Find information from local documentation in GNU Info.

Chapter 5, Creating, Viewing, and Editing Text Files

Create, view, and edit text files from command output or in a text editor.

- Save command output or errors to a file with shell redirection, and process command output through multiple command-line programs with pipes.
- Create and edit text files using the **vim** editor.
- Use shell variables to help run commands, and edit Bash startup scripts to set shell and environment variables to modify the behavior of the shell and programs run from the shell.

Chapter 6, Managing Local Users and Groups

Create, manage, and delete local users and groups and administer local password policies.

- Describe the purpose of users and groups on a Linux system.
- Switch to the superuser account to manage a Linux system, and grant other users superuser access using the **sudo** command.
- Create, modify, and delete locally defined user accounts.
- Create, modify, and delete locally defined group accounts.
- Set a password management policy for users, and manually lock and unlock user accounts.

Chapter 7, Controlling Access to Files

Set Linux file-system permissions on files and to interpret the security effects of different permission settings.

- List the file system permissions on files and directories, and interpret the effect of those permissions on access by users and groups.
- Change the permissions and ownership of files using command-line tools.
- Control the default permissions of new files created by users, explain the effect of special permissions, and use special permissions and default permissions to set the group owner of files created in a particular directory.

Chapter 8, Monitoring and Managing Linux Processes

Evaluate and control processes running on a Red Hat Enterprise Linux system.

- Get information about programs running on the system so that you can determine status, resource use, and ownership, so you can control them.
- Use Bash job control to manage multiple processes started from the same terminal session.
- Control and terminate processes that are not associated with your shell, and forcibly end user sessions and processes.
- Describe what load average is and determine processes responsible for high resource use on a server.

Chapter 9, Controlling Services and Daemons

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 10, Configuring and Securing SSH

Configure secure command-line service on remote systems, using OpenSSH.

- Log in to a remote system and run commands using **ssh**.
- Configure key-based authentication for a user account to log in to remote systems securely without a password.
- Restrict direct logins as root and disable password-based authentication for the OpenSSH service.

Chapter 11, Analyzing and Storing Logs

Locate and accurately interpret logs of system events for troubleshooting purposes.

- Describe the basic logging architecture used by Red Hat Enterprise Linux to record events.
- Interpret events in relevant syslog files to troubleshoot problems or review system status.
- Find and interpret entries in the system journal to troubleshoot problems or review system status.
- Configure the system journal to preserve the record of events when a server is rebooted.
- Maintain accurate time synchronization using NTP and configure the time zone to ensure correct time stamps for events recorded by the system journal and logs.

Chapter 12, Managing Networking

Configure network interfaces and settings on Red Hat Enterprise Linux servers.

- Describe fundamental concepts of network addressing and routing for a server.
- Test and inspect current network configuration with command-line utilities.
- Manage network settings and devices using **nmcli**.
- Modify network settings by editing configuration files.
- Configure a server's static host name and its name resolution, and test the results.

Chapter 13, Archiving and Transferring Files

Archive and copy files from one system to another.

- Archive files and directories into a compressed file using tar, and extract the contents of an existing tar archive.
- Transfer files to or from a remote system securely using SSH.
- Synchronize the contents of a local file or directory with a copy on a remote server.

Chapter 14, Installing and Updating Software Packages

Download, install, update, and manage software packages from Red Hat and Yum package repositories.

- Register a system to your Red Hat account and assign it entitlements for software updates and support services using Red Hat Subscription Management.
- Explain how software is provided as RPM packages, and investigate the packages installed on the system with Yum and RPM.
- Find, install, and update software packages using the **yum** command.
- Enable and disable use of Red Hat or third-party Yum repositories by a server.
- Explain how modules allow installation of specific versions of software, list, enable, and switch module streams, and install and update packages from a module.

Chapter 15, Accessing Linux File Systems

Access, inspect, and use existing file systems on storage attached to a Linux server.

- Explain what a block device is, interpret the file names of storage devices, and identify the storage device used by the file system for a particular directory or file.
- Access file systems by attaching them to a directory in the file system hierarchy.
- Search for files on mounted file systems using the **find** and **locate** commands.

Chapter 16, Analyzing Servers and Getting Support

Investigate and resolve issues in the web-based management interface, getting support from Red Hat to help solve problems.

- Activate the Web Console management interface to remotely manage and monitor the performance of a Red Hat Enterprise Linux server.
- Describe key resources available through the Red Hat Customer Portal, and find information from Red Hat documentation and the Knowledgebase.
- Analyze servers for issues, remediate or resolve them, and confirm the solution with Red Hat Insights.

▶ Lab

Managing Files from the Command Line

In this review, you will manage files, redirect a specific set of lines from a text file to another file and edit the text files.

Outcomes

You should be able to:

- Manage files from the command line.
- Display a certain number of lines from text files and redirect the output to another file.
- Edit text files.

Before You Begin

Copy any files or work you wish to keep to other systems before resetting. Reset the **workstation**, **servera**, and **serverb** systems now. Wait until the **workstation**, **servera**, and **serverb** systems are started.

Log in to **workstation** as **student** using **student** as the password.

On **workstation**, run **lab rhcsa-rh124-review1 start** to start the comprehensive review. This script creates the necessary files to set up the environment correctly.

```
[student@workstation ~]$ lab rhcsa-rh124-review1 start
```

Instructions

Accomplish the following tasks on **serverb** to complete the exercise.

- Create a new directory called **/home/student/grading**.
- Create three empty files in the **/home/student/grading** directory named **grade1**, **grade2**, and **grade3**.
- Capture the first five lines of the **/home/student/bin/manage-files** file in the **/home/student/grading/manage-files.txt** file.
- Append the last three lines of **/home/student/bin/manage-files** to the file **/home/student/grading/manage-files.txt**. You must not overwrite any text already in the file **/home/student/grading/manage-files.txt**.
- Copy **/home/student/grading/manage-files.txt** to **/home/student/grading/manage-files-copy.txt**.
- Edit the file **/home/student/grading/manage-files-copy.txt** so that there should be two sequential lines of text reading **Test JJ**.
- Edit the file **/home/student/grading/manage-files-copy.txt** so that the **Test HH** line of text must not exist in the file.
- Edit the file **/home/student/grading/manage-files-copy.txt** so that the line **A new line** should exist between the line reading **Test BB** and the line reading **Test CC**.

- Create a hard link named **/home/student/hardlink** to the file **/home/student/grading/grade1**. You will need to do this after creating the empty file **/home/student/grading/grade1** as specified above.
- Create a soft link named **/home/student/softlink** to the file **/home/student/grading/grade2**.
- Save the output of a command that lists the contents of the **/boot** directory to the file **/home/student/grading/longlisting.txt**. The output should be a “long listing” that includes file permissions, owner and group owner, size, and modification date of each file.

Evaluation

On **workstation**, run the **lab rhcsa-rh124-review1 grade** command to confirm success of this exercise.

```
[student@workstation ~]$ lab rhcsa-rh124-review1 grade
```

Finish

On **workstation**, run **lab rhcsa-rh124-review1 finish** to complete the comprehensive review. This script deletes the files and directories created during the start of the comprehensive review and ensures that the environment on **serverb** is clean.

```
[student@workstation ~]$ lab rhcsa-rh124-review1 finish
```

This concludes the comprehensive review.

► Solution

Managing Files from the Command Line

In this review, you will manage files, redirect a specific set of lines from a text file to another file and edit the text files.

Outcomes

You should be able to:

- Manage files from the command line.
- Display a certain number of lines from text files and redirect the output to another file.
- Edit text files.

Before You Begin

Copy any files or work you wish to keep to other systems before resetting. Reset the **workstation**, **servera**, and **serverb** systems now. Wait until the **workstation**, **servera**, and **serverb** systems are started.

Log in to **workstation** as **student** using **student** as the password.

On **workstation**, run **lab rhcsa-rh124-review1 start** to start the comprehensive review. This script creates the necessary files to set up the environment correctly.

```
[student@workstation ~]$ lab rhcsa-rh124-review1 start
```

Instructions

Accomplish the following tasks on **serverb** to complete the exercise.

- Create a new directory called **/home/student/grading**.
- Create three empty files in the **/home/student/grading** directory named **grade1**, **grade2**, and **grade3**.
- Capture the first five lines of the **/home/student/bin/manage-files** file in the **/home/student/grading/manage-files.txt** file.
- Append the last three lines of **/home/student/bin/manage-files** to the file **/home/student/grading/manage-files.txt**. You must not overwrite any text already in the file **/home/student/grading/manage-files.txt**.
- Copy **/home/student/grading/manage-files.txt** to **/home/student/grading/manage-files-copy.txt**.
- Edit the file **/home/student/grading/manage-files-copy.txt** so that there should be two sequential lines of text reading **Test JJ**.
- Edit the file **/home/student/grading/manage-files-copy.txt** so that the **Test HH** line of text must not exist in the file.
- Edit the file **/home/student/grading/manage-files-copy.txt** so that the line **A new line** should exist between the line reading **Test BB** and the line reading **Test CC**.

- Create a hard link named **/home/student/hardlink** to the file **/home/student/grading/grade1**. You will need to do this after creating the empty file **/home/student/grading/grade1** as specified above.
- Create a soft link named **/home/student/softlink** to the file **/home/student/grading/grade2**.
- Save the output of a command that lists the contents of the **/boot** directory to the file **/home/student/grading/longlisting.txt**. The output should be a “long listing” that includes file permissions, owner and group owner, size, and modification date of each file.

1. Create a new directory called **/home/student/grading**.

1.1. From **workstation**, open an SSH session to **serverb** as **student**.

```
[student@workstation ~]$ ssh student@serverb  
...output omitted...  
[student@serverb ~]$
```

1.2. Use the **mkdir** command to create the **/home/student/grading** directory.

```
[student@serverb ~]$ mkdir grading
```

As you ran the preceding command while in the home directory of the **student** user, you did not specify the absolute path to the **grading** directory while creating it.

2. Create three empty files in the **/home/student/grading** directory named **grade1**, **grade2**, and **grade3**.

2.1. Use the **touch** command to create the empty files called **grade1**, **grade2**, and **grade3** in the **/home/student/grading** directory. Apply the brace expansion shell feature to create all three files with a single **touch** command.

```
[student@serverb ~]$ touch grading/grade{1,2,3}
```

2.2. Use the **ls** command to verify that the **grade1**, **grade2**, and **grade3** files exist under the directory **/home/student/grading**.

```
[student@serverb ~]$ ls grading/  
grade1 grade2 grade3
```

3. Capture the first five lines of the **/home/student/bin/manage-files** file in the **/home/student/grading/manage-files.txt** file.

3.1. Use the **head** command to view the first five lines of the file **/home/student/bin/manage-files** and redirect the output to the file **/home/student/grading/manage-files.txt**.

```
[student@serverb ~]$ head -5 bin/manage-files > grading/manage-files.txt
```

The preceding command uses the single redirection symbol (**>**) to save the command output to **/home/student/grading/manage-files.txt** so that any existing content in the file gets overwritten.

- 3.2. Verify that the file **/home/student/grading/manage-files.txt** contains the following text.

```
Test AA  
Test BB  
Test CC  
Test DD  
Test EE
```

4. Append the last three lines of **/home/student/bin/manage-files** to the file **/home/student/grading/manage-files.txt**. You must not overwrite any text already in the file **/home/student/grading/manage-files.txt**.

- 4.1. Use the **tail** command to view the last three lines of the file **/home/student/bin/manage-files** and append the output to **/home/student/grading/manage-files.txt**.

```
[student@serverb ~]$ tail -3 bin/manage-files >> grading/manage-files.txt
```

The preceding command uses the double redirection symbol (**>>**) to append the output to **/home/student/grading/manage-files.txt** so that the existing contents in the file are preserved.

- 4.2. Verify that the file **/home/student/grading/manage-files.txt** contains the following text.

```
Test AA  
Test BB  
Test CC  
Test DD  
Test EE  
Test HH  
Test II  
Test JJ
```

5. Copy the **/home/student/grading/manage-files.txt** file to **/home/student/grading/manage-files-copy.txt**.

- 5.1. Use the **cd** command to navigate to the directory **/home/student/grading**.

```
[student@serverb ~]$ cd grading/  
[student@serverb grading]$
```

- 5.2. Use the **cp** command to copy the **/home/student/grading/manage-files.txt** file to **/home/student/grading/manage-files-copy.txt**.

```
[student@serverb grading]$ cp manage-files.txt manage-files-copy.txt
```

- 5.3. Navigate back to the home directory of the user **student**.

```
[student@serverb grading]$ cd
[student@serverb ~]$
```

6. Edit the file **/home/student/grading/manage-files-copy.txt** so that there should be two sequential lines of text reading **Test JJ**.

- 6.1. Use the **vim** text editor to open the **/home/student/grading/manage-files-copy.txt** file.

```
[student@serverb ~]$ vim grading/manage-files-copy.txt
```

- 6.2. From the command mode in **vim**, scroll down to the line that has the **Test JJ** line of text. Press the **y** key twice on your keyboard to copy the line of text and press the **p** key to paste it below the cursor. Type **:wq** to save the changes and quit **vim**. Verify that the **/home/student/grading/manage-files-copy.txt** file contains the following text.

```
Test AA
Test BB
Test CC
Test DD
Test EE
Test HH
Test II
Test JJ
Test JJ
```

Notice that the preceding content includes two copies of the **Test JJ** line of text.

7. Edit the file **/home/student/grading/manage-files-copy.txt** so that the **Test HH** line of text must not exist in the file.

- 7.1. Use the **vim** text editor to open the **/home/student/grading/manage-files-copy.txt** file.

```
[student@serverb ~]$ vim grading/manage-files-copy.txt
```

- 7.2. From the command mode in **vim**, scroll down to the line that has the **Test HH** line of text. Press the **d** key twice on your keyboard to delete the line of text. Type **:wq** to save the changes and quit **vim**. Verify that the **/home/student/grading/manage-files-copy.txt** file contains the following text.

```
Test AA
Test BB
Test CC
Test DD
Test EE
Test II
Test JJ
Test JJ
```

Notice that the preceding content does not include the **Test HH** line of text.

8. Edit the file `/home/student/grading/manage-files-copy.txt` so that the line **A new line** exists between the line reading **Test BB** and the line reading **Test CC**.

- 8.1. Use the `vim` text editor to open the `/home/student/grading/manage-files-copy.txt` file.

```
[student@serverb ~]$ vim grading/manage-files-copy.txt
```

- 8.2. From the command mode in `vim`, scroll down to the line that has the **Test CC** line of text. Press the **i** key on the keyboard to switch to the insert mode while keeping the cursor at the beginning of the **Test CC** line of text. From the insert mode, press the **Enter** key on the keyboard to create a blank line above the cursor. Use the up arrow to navigate to the blank line and create the **A new line** line of text. Press the **Esc** key on the keyboard to switch back to the command mode. Type `:wq` to save the changes and quit `vim`. Verify that the `/home/student/grading/manage-files-copy.txt` file contains the following text.

```
Test AA
Test BB
A new line
Test CC
Test DD
Test EE
Test II
Test JJ
Test JJ
```

Notice that the preceding content includes the **A new line** line of text.

9. Create a hard link named `/home/student/hardlink` to the file `/home/student/grading/grade1`.

- 9.1. Use the `ln` command to create the hard link named `/home/student/hardlink` to the file `/home/student/grading/grade1`. You will need to do this after creating the empty file `/home/student/grading/grade1` as specified above.

```
[student@serverb ~]$ ln grading/grade1 hardlink
```

- 9.2. Use the `ls -l` command to view the link count of the `/home/student/grading/grade1` file.

```
[student@serverb ~]$ ls -l grading/grade1
-rw-rw-r--. 2 student student 0 Mar  6 16:45 grading/grade1
```

10. Create a soft link named `/home/student/softlink` to the file `/home/student/grading/grade2`.

- 10.1. Use the `ln -s` command to create the soft link named `/home/student/softlink` to the file `/home/student/grading/grade2`.

```
[student@serverb ~]$ ln -s grading/grade2 softlink
```

- 10.2. Use the `ls -l` command to view the properties of the `/home/student/softlink` soft link.

```
[student@serverb ~]$ ls -l softlink  
lrwxrwxrwx. 1 student student 14 Mar 6 17:58 softlink -> grading/grade2
```

11. Save the output of a command that lists the contents of the **/boot** directory to the file **/home/student/grading/longlisting.txt**. The output should be a “long listing” that includes file permissions, owner and group owner, size, and modification date of each file.
- 11.1. Use the **ls -l** command to view the contents of the directory **/boot** in the “long listing” format and redirect the output to the file **/home/student/grading/longlisting.txt**.

```
[student@serverb ~]$ ls -l /boot > grading/longlisting.txt
```

- 11.2. Log out of **serverb**.

```
[student@serverb ~]$ exit  
logout  
Connection to serverb closed.
```

Evaluation

On **workstation**, run the **lab rhcsa-rh124-review1 grade** command to confirm success of this exercise.

```
[student@workstation ~]$ lab rhcsa-rh124-review1 grade
```

Finish

On **workstation**, run **lab rhcsa-rh124-review1 finish** to complete the comprehensive review. This script deletes the files and directories created during the start of the comprehensive review and ensures that the environment on **serverb** is clean.

```
[student@workstation ~]$ lab rhcsa-rh124-review1 finish
```

This concludes the comprehensive review.

▶ Lab

Managing Users and Groups, Permissions and Processes

In this review, you will manage user and group accounts, set permissions on files and directories, and manage processes.

Outcomes

You should be able to:

- Manage users and groups.
- Set permissions on files and directories.
- Remove processes that are consuming too much CPU.

Before You Begin

Log in to **workstation** as **student** using **student** as the password.

On **workstation**, run **lab rhcsa-rh124-review2 start** to start the comprehensive review. This script runs a process that consumes the maximum CPU resources and creates the necessary files to set up the environment correctly.

```
[student@workstation ~]$ lab rhcsa-rh124-review2 start
```

Instructions

Accomplish the following tasks on **serverb** to complete the exercise.

- Terminate the process that is currently using the most CPU time.
- Create a new group called **database** that has the GID **50000**.
- Create a new user called **dbuser1** that uses the group **database** as one of its secondary groups. The initial password of **dbuser1** should be set to **redhat**. Configure the user **dbuser1** to force a password change on its first login. The user **dbuser1** should be able to change its password after **10** days since the day of the password change. The password of **dbuser1** should expire in **30** days since the last day of the password change.
- Configure the user **dbuser1** to use **sudo** to run any command as the superuser.
- Configure the user **dbuser1** to have a default umask of **007**.
- The permissions on **/home/student/grading/review2** should allow the group members of **database** and the user **student** to access the directory and create contents in it. All other users should have read and execute permissions on the directory. Also, ensure that users are only allowed to delete files they own from **/home/student/grading/review2** and not files belonging to others.

Evaluation

On **workstation**, run the **lab rhcsa-rh124-review2 grade** command to confirm success of this exercise.

```
[student@workstation ~]$ lab rhcsa-rh124-review2 grade
```

Finish

On **workstation**, run **lab rhcsa-rh124-review2 finish** to complete the comprehensive review. This script terminates the process and deletes the files and directories created during the start of the comprehensive review and ensures that the environment on **serverb** is clean.

```
[student@workstation ~]$ lab rhcsa-rh124-review2 finish
```

This concludes the comprehensive review.

► Solution

Managing Users and Groups, Permissions and Processes

In this review, you will manage user and group accounts, set permissions on files and directories, and manage processes.

Outcomes

You should be able to:

- Manage users and groups.
- Set permissions on files and directories.
- Remove processes that are consuming too much CPU.

Before You Begin

Log in to **workstation** as **student** using **student** as the password.

On **workstation**, run **lab rhcsa-rh124-review2 start** to start the comprehensive review. This script runs a process that consumes the maximum CPU resources and creates the necessary files to set up the environment correctly.

```
[student@workstation ~]$ lab rhcsa-rh124-review2 start
```

Instructions

Accomplish the following tasks on **serverb** to complete the exercise.

- Terminate the process that is currently using the most CPU time.
- Create a new group called **database** that has the GID **50000**.
- Create a new user called **dbuser1** that uses the group **database** as one of its secondary groups. The initial password of **dbuser1** should be set to **redhat**. Configure the user **dbuser1** to force a password change on its first login. The user **dbuser1** should be able to change its password after **10** days since the day of the password change. The password of **dbuser1** should expire in **30** days since the last day of the password change.
- Configure the user **dbuser1** to use **sudo** to run any command as the superuser.
- Configure the user **dbuser1** to have a default umask of **007**.
- The permissions on **/home/student/grading/review2** should allow the group members of **database** and the user **student** to access the directory and create contents in it. All other users should have read and execute permissions on the directory. Also, ensure that users are only allowed to delete files they own from **/home/student/grading/review2** and not files belonging to others.

1. Terminate the process that is currently using the most CPU time.

- From **workstation**, open an SSH session to **serverb** as **student**.

```
[student@workstation ~]$ ssh student@serverb
...output omitted...
[student@serverb ~]$
```

- Use the **top** command to view the real-time system status.

```
[student@serverb ~]$ top
```

- From the interactive interface of **top**, pay attention to the **%CPU** column and confirm that there is a process called **dd**, consuming the most CPU resources.

```
...output omitted...
PID USER PR NI VIRT RES SHR S %CPU %MEM TIME+ COMMAND
2303 student 20 0 217048 944 876 R 99.7 0.1 100:11.64 dd
...output omitted...
```

Notice the process **dd** with the PID **2303** in the preceding output, which is consuming the majority of CPU resources, at **99.7%**. The PID and the percentage of CPU resource consumption may vary in your system.

- From the interactive interface of **top**, type **k** to kill the process **dd** with PID **2303**, as you determined in the preceding step. If the default PID shown in the prompt matches that of the process consuming the majority of CPU resources, press the **Enter** key on the keyboard. If it does not match, specify the PID interactively.

```
...output omitted...
PID to signal/kill [default pid = 2303] Enter
...output omitted...
```

- Use the default signal **SIGTERM** to terminate the process.

```
...output omitted...
Send pid 2833 signal [15/sigterm] Enter
...output omitted...
```

- From the interactive interface, press the **q** key on the keyboard to quit **top**.

- Create a new group called **database** with GID **50000**.

- Switch to the user **root**.

```
[student@serverb ~]$ sudo su -
[sudo] password for student: student
[root@serverb ~]#
```

- Use the **groupadd** command to create a new group called **database** with GID **50000**.

```
[root@serverb ~]# groupadd -g 50000 database
```

3. Create a new user called **dbuser1** with group **database** as one of its secondary groups. Set the initial password of **dbuser1** to **redhat**. Configure the user **dbuser1** to force a password change upon first login. The user **dbuser1** should be able to change its password after **10** days since the last day of the password change. The password of **dbuser1** should expire in **30** days since the last day of the password change.

- 3.1. Use the **useradd** command to create a new user called **dbuser1** that uses the group **database** as one of its secondary groups.

```
[root@serverb ~]# useradd -G database dbuser1
```

- 3.2. Use the **passwd** command to set the password of **dbuser1** to **redhat**.

```
[root@serverb ~]# passwd dbuser1
Changing password for user dbuser1.
New password: redhat
BAD PASSWORD: The password is shorter than 8 characters
Retype new password: redhat
passwd: all authentication tokens updated successfully.
```

- 3.3. Use the **chage** command to force **dbuser1** to change its password on first login.

```
[root@serverb ~]# chage -d 0 dbuser1
```

- 3.4. Use the **chage** command to set the minimum age of the password of **dbuser1** to **10** days.

```
[root@serverb ~]# chage -m 10 dbuser1
```

- 3.5. Use the **chage** command to set the maximum age of the password of **dbuser1** to **30** days.

```
[root@serverb ~]# chage -M 30 dbuser1
```

4. Create the file **/etc/sudoers.d/dbuser1** to configure **dbuser1** so that the user can use **sudo** to run any command as the superuser. You may use the **vim /etc/sudoers.d/dbuser1** command to create the file. The **/etc/sudoers.d/dbuser1** should contain the following content.

- 4.1.

```
dbuser1 ALL=(ALL) ALL
```

5. Configure the user **dbuser1** to have a default umask of **007**.

- 5.1. Switch to the user **dbuser1**.

```
[root@serverb ~]# su - dbuser1
[dbuser1@serverb ~]$
```

- 5.2. Append the line **umask 007** to the files **/home/dbuser1/.bash_profile** and **/home/dbuser1/.bashrc**.

```
[dbuser1@serverb ~]$ echo "umask 007" >> .bash_profile
[dbuser1@serverb ~]$ echo "umask 007" >> .bashrc
```

- 5.3. Exit the **dbuser1** user's shell.

```
[dbuser1@serverb ~]$ exit
logout
[root@serverb ~]#
```

6. Create a new directory called **/home/student/grading/review2** with **student** and **database** as its owning user and group respectively. Configure the permissions on that directory so that any new file in it inherits **database** as its owning group irrespective to the creating user. The permissions on **/home/student/grading/review2** should allow the group members of **database** and the user **student** to access the directory and create contents in it. All other users should have read and execute permissions on the directory. Also, ensure that the users are only allowed to delete the files, they own, from **/home/student/grading/review2** and not others' files.

- 6.1. Use the **mkdir** command to create **/home/student/grading/review2**.

```
[root@serverb ~]# mkdir /home/student/grading/review2
```

- 6.2. On the **/home/student/grading/review2**, use the **chown** command to set **student** and **database** as the owning user and group respectively.

```
[root@serverb ~]# chown student:database /home/student/grading/review2
```

- 6.3. Use the **chmod** command to apply the SetGID special permission on **/home/student/grading/review2**.

```
[root@serverb ~]# chmod g+s /home/student/grading/review2
```

- 6.4. Use the **chmod** command to apply the permission mode **775** on **/home/student/grading/review2**.

```
[root@serverb ~]# chmod 775 /home/student/grading/review2
```

- 6.5. Use the **chmod** command to apply the stickybit special permission on **/home/student/grading/review2**.

```
[root@serverb ~]# chmod o+t /home/student/grading/review2
```

- 6.6. Exit the **root** user's shell.

```
[root@serverb ~]# exit
logout
[student@serverb ~]$
```

- 6.7. Log out of **serverb**.

```
[student@serverb ~]$ exit  
logout  
Connection to serverb closed.  
[student@workstation ~]$
```

Evaluation

On **workstation**, run the **lab rhcsa-rh124-review2 grade** command to confirm success of this exercise.

```
[student@workstation ~]$ lab rhcsa-rh124-review2 grade
```

Finish

On **workstation**, run **lab rhcsa-rh124-review2 finish** to complete the comprehensive review. This script terminates the process and deletes the files and directories created during the start of the comprehensive review and ensures that the environment on **serverb** is clean.

```
[student@workstation ~]$ lab rhcsa-rh124-review2 finish
```

This concludes the comprehensive review.

▶ Lab

Configuring and Managing a Server

In this review, you will configure, secure, and use SSH service to access remote machine, configure **rsyslog** service, archive local files, transfer local files to remote machine, and manage packages using **yum**.

Outcomes

You should be able to:

- Create a new SSH key pair.
- Disable SSH logins as **root** user.
- Disable SSH logins using password.
- Update the time zone of a server.
- Install packages and package modules using **yum**.
- Archive local files for backup.
- Transfer local files to remote machine.

Before You Begin

Log in to **workstation** as **student** using **student** as the password.

On **workstation**, run **lab rhcsa-rh124-review3 start** to start the comprehensive review. This script creates the necessary files to set up the environment correctly.

```
[student@workstation ~]$ lab rhcsa-rh124-review3 start
```

Instructions

Accomplish the following tasks on **serverb** to complete the exercise.

- Generate SSH keys for the user **student** on **serverb**. Do not protect the private key with a passphrase. The private and public key files should be named **/home/student/.ssh/review3_key** and **/home/student/.ssh/review3_key.pub** respectively.
- On **servera**, configure the user **student** to accept logins authenticated by the SSH key pair you created for the user **student** on **serverb**. The user **student** on **serverb** should be able to log in to **servera** using SSH without entering a password.
- On **serverb**, configure the **sshd** service to prevent users from logging in as **root** via SSH.
- On **serverb**, configure the **sshd** service to prevent users from using their passwords to log in. Users should still be able to authenticate logins using an SSH key pair.
- Create a tar archive named **/tmp/log.tar** containing the contents of **/var/log** on **serverb**. Remotely transfer the tar archive to **/tmp** directory on **servera**, authenticating as **student** using the **student** user's private key of the SSH key pair.

- Configure the **rsyslog** service on **serverb** to log all messages it receives that have the priority level of **debug** or higher to the file **/var/log/grading-debug**. This configuration should be set in an **/etc/rsyslog.d/grading-debug.conf** file, which you need to create.
- Install the **zsh** package, available in the BaseOS repository, on **serverb**.
- Enable the default module stream for the module **python36** and install all provided packages from that stream on **serverb**.
- Set the time zone of **serverb** to **Asia/Kolkata**.

Evaluation

On **workstation**, run the **lab rhcsa-rh124-review3 grade** command to confirm success of this exercise.

```
[student@workstation ~]$ lab rhcsa-rh124-review3 grade
```

Finish

On **workstation**, run **lab rhcsa-rh124-review3 finish** to complete the comprehensive review. This script deletes the files and directories created during the start of the comprehensive review and ensures that the environment on **serverb** is clean.

```
[student@workstation ~]$ lab rhcsa-rh124-review3 finish
```

This concludes the comprehensive review.

► Solution

Configuring and Managing a Server

In this review, you will configure, secure, and use SSH service to access remote machine, configure **rsyslog** service, archive local files, transfer local files to remote machine, and manage packages using **yum**.

Outcomes

You should be able to:

- Create a new SSH key pair.
- Disable SSH logins as **root** user.
- Disable SSH logins using password.
- Update the time zone of a server.
- Install packages and package modules using **yum**.
- Archive local files for backup.
- Transfer local files to remote machine.

Before You Begin

Log in to **workstation** as **student** using **student** as the password.

On **workstation**, run **lab_rhcsa-rh124-review3 start** to start the comprehensive review. This script creates the necessary files to set up the environment correctly.

```
[student@workstation ~]$ lab_rhcsa-rh124-review3 start
```

Instructions

Accomplish the following tasks on **serverb** to complete the exercise.

- Generate SSH keys for the user **student** on **serverb**. Do not protect the private key with a passphrase. The private and public key files should be named **/home/student/.ssh/review3_key** and **/home/student/.ssh/review3_key.pub** respectively.
- On **servera**, configure the user **student** to accept logins authenticated by the SSH key pair you created for the user **student** on **serverb**. The user **student** on **serverb** should be able to log in to **servera** using SSH without entering a password.
- On **serverb**, configure the **sshd** service to prevent users from logging in as **root** via SSH.
- On **serverb**, configure the **sshd** service to prevent users from using their passwords to log in. Users should still be able to authenticate logins using an SSH key pair.
- Create a tar archive named **/tmp/log.tar** containing the contents of **/var/log** on **serverb**. Remotely transfer the tar archive to **/tmp** directory on **servera**, authenticating as **student** using the **student** user's private key of the SSH key pair.

- Configure the **rsyslog** service on **serverb** to log all messages it receives that have the priority level of **debug** or higher to the file **/var/log/grading-debug**. This configuration should be set in an **/etc/rsyslog.d/grading-debug.conf** file, which you need to create.
- Install the **zsh** package, available in the BaseOS repository, on **serverb**.
- Enable the default module stream for the module **python36** and install all provided packages from that stream on **serverb**.
- Set the time zone of **serverb** to **Asia/Kolkata**.

- Generate SSH keys for the user **student** on **serverb**. Do not protect the private key with a passphrase. The private and public key files should be named **/home/student/.ssh/review3_key** and **/home/student/.ssh/review3_key.pub** respectively.

- From **workstation**, open an SSH session to **serverb** as **student**.

```
[student@workstation ~]$ ssh student@serverb
...output omitted...
[student@serverb ~]$
```

- Use the **ssh-keygen** command to generate the SSH keys for the user **student**.

```
[student@serverb ~]$ ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/home/student/.ssh/id_rsa): /home/
student/.ssh/review3_key
Enter passphrase (empty for no passphrase): Enter
Enter same passphrase again: Enter
Your identification has been saved in /home/student/.ssh/review3_key.
Your public key has been saved in /home/student/.ssh/review3_key.pub.
The key fingerprint is:
SHA256:Uqefehw+vRfm94fQZDoz/6IfNYSLK/OpiQ4n6lrKIbY student@serverb.lab.example.com
The key's randomart image is:
+---[RSA 2048]---+
|                               |
|                               |
|                               . |
|                               .. |
|                               . o . = |
|                               . S . * ..|
|                               . ...B +..|
|.o . o . =o+ 0.o |
|+= . + ..X o * .o|
| Eoo .o.+.+o=.+=|
+---[SHA256]---+
```

- On **servera**, configure the user **student** to accept logins authenticated by the SSH key pair you created for the user **student** on **serverb**. The user **student** on **serverb** should be able to log in to **servera** using SSH without entering a password.

- Use the **ssh-copy-id** command to export the public key **/home/student/.ssh/review3_key.pub** from **servera** to **serverb**.

```
[student@serverb ~]$ ssh-copy-id -i .ssh/review3_key.pub student@servera
/usr/bin/ssh-copy-id: INFO: Source of key(s) to be installed: ".ssh/review3.pub"
/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@servera's password: student

Number of key(s) added: 1

Now try logging into the machine, with: "ssh 'student@servera'"
and check to make sure that only the key(s) you wanted were added.
```

- 2.2. Use the **ssh** command to confirm that you can log in to **servera** from **serverb** as **student** using the SSH private key **/home/student/.ssh/review3_key** without being prompted for the password.

```
[student@serverb ~]$ ssh -i .ssh/review3_key student@servera
...output omitted...
[student@servera ~]$
```

- 2.3. Log out of **servera**.

```
[student@servera ~]$ exit
logout
Connection to servera closed.
[student@serverb ~]$
```

3. On **serverb**, configure the **sshd** service to prevent users from logging in as **root** with SSH.

- 3.1. Set the parameter **PermitRootLogin** to **no** in the **/etc/ssh/sshd_config**. You may use the command **sudo vim /etc/ssh/sshd_config** to edit the configuration file.
- 3.2. Reload the **sshd** service.

```
[student@serverb ~]$ sudo systemctl reload sshd.service
```

4. On **serverb**, configure the **sshd** service to prevent users from using their passwords to log in. Users should still be able to authenticate logins using their private key of the SSH key pair.

- 4.1. Set the parameter **PasswordAuthentication** to **no** in the **/etc/ssh/sshd_config**. You may use the command **sudo vim /etc/ssh/sshd_config** to edit the configuration file.
- 4.2. Use the **sudo systemctl** command to reload the **sshd** service.

```
[student@serverb ~]$ sudo systemctl reload sshd.service
```

5. Create a tar archive named **/tmp/log.tar** containing the contents of **/var/log** on **serverb**. Remotely transfer the tar archive to the directory **/tmp** on **servera**,

authenticating as **student** using **/home/student/.ssh/review3_key** as the **student** user's private key of the SSH key pair for authentication.

- 5.1. Use the **sudo tar** command to create an archive named **/tmp/log.tar** as the superuser containing the contents of **/var/log**.

```
[student@serverb ~]$ sudo tar -cvf /tmp/log.tar /var/log
[sudo] password for student: student
...output omitted...
```

- 5.2. Use the **scp** command to remotely transfer the archive file **/tmp/log.tar** to the directory **/tmp** on **servera**. Specify **/home/student/.ssh/review3_key** as the private key of the SSH key pair.

```
[student@serverb ~]$ scp -i .ssh/review3_key /tmp/log.tar student@servera:/tmp
log.tar                                100%   14MB  57.4MB/S   00:00
```

6. Configure the **rsyslog** service on **serverb** to log all messages it receives that have the priority level of **debug** or higher to the file **/var/log/grading-debug**. This configuration should be set in an **/etc/rsyslog.d/grading-debug.conf** file which you should create.

- 6.1. Create the file **/etc/rsyslog.d/grading-debug.conf** with the following content. You may use the **sudo vim /etc/rsyslog.d/grading-debug.conf** to create the file.

```
*.debug /var/log/grading-debug
```

- 6.2. Use the **sudo systemctl** command to restart the **rsyslog** service.

```
[student@serverb ~]$ sudo systemctl restart rsyslog.service
```

- 6.3. Use the **logger** command to generate the log message **Debug Testing** having priority **debug**.

```
[student@serverb ~]$ logger -p debug Debug Testing
```

- 6.4. Confirm that the log message **Debug Testing** is saved in the **/var/log/grading-debug** file.

```
[student@serverb ~]$ sudo tail /var/log/grading-debug
...output omitted...
Mar 12 09:55:23 serverb student[32383]: Debug Testing
```

7. Use the **sudo yum** command to install the **zsh** package, available in the BaseOS repository, on **serverb**.

7.1.

```
[student@serverb ~]$ sudo yum install zsh
...output omitted...
Is this ok [y/N]: y
...output omitted...
Installed:
zsh-5.5.1-6.el8.x86_64
Complete!
```

8. Use the **yum** command to enable the default module stream for the module **python36** and install all the provided packages from that stream, on **serverb**.

8.1.

```
[student@serverb ~]$ sudo yum module install python36
...output omitted...
Is this ok [y/N]: y
...output omitted...
Installed:
python36-3.6.6-18.module+el8+2339+1a6691f8.x86_64          python3-
pip-9.0.3-13.el8.noarch

Complete!
```

9. Set the timezone of **serverb** to **Asia/Kolkata**.

- 9.1. Use the **sudo timedatectl** command to set the timezone of **serverb** to **Asia/Kolkata**.

```
[student@serverb ~]$ sudo timedatectl set-timezone Asia/Kolkata
```

- 9.2. Log out of **serverb**.

```
[student@serverb ~]$ exit
logout
Connection to serverb closed.
[student@workstation ~]$
```

Evaluation

On **workstation**, run the **lab rhcsa-rh124-review3 grade** command to confirm success of this exercise.

```
[student@workstation ~]$ lab rhcsa-rh124-review3 grade
```

Finish

On **workstation**, run **lab rhcsa-rh124-review3 finish** to complete the comprehensive review. This script deletes the files and directories created during the start of the comprehensive review and ensures that the environment on **serverb** is clean.

```
[student@workstation ~]$ lab rhcsa-rh124-review3 finish
```

This concludes the comprehensive review.

▶ Lab

Managing Networks

In this review, you will configure and test network connectivity.

Outcomes

You should be able to:

- Configure the network settings.
- Test network connectivity.
- Set a static host name for the system.
- Use locally resolvable canonical host names to connect to systems.

Before You Begin

Log in to **workstation** as **student** using **student** as the password.

On **workstation**, run **lab rhcsa-rh124-review4 start** to start the comprehensive review. This script creates the necessary files to set up the environment correctly.

```
[student@workstation ~]$ lab rhcsa-rh124-review4 start
```

Instructions

Accomplish the following tasks on **serverb** to complete the exercise.



Warning

It is a useful practice to make network changes from the server console, whether locally or through remote console access hardware. When using **ssh** to adjust networking settings, a mistaken command can hang or lock out your session. Network configuration corrections must then be made through the console.

In the web page that controls your lab environment, click the **OPEN CONSOLE** button for **serverb**. A tab will open in your browser with the console session for **serverb**. Log in as user **student** at the prompt.

- Determine the name of the Ethernet interface and its active connection profile on **serverb**.
- On **serverb**, create a new connection profile called **static** for the available Ethernet interface that statically sets network settings and does not use DHCP. Use the settings in the following table.

IPv4 address	172.25.250.111
Netmask	255.255.255.0

Gateway	172.25.250.254
DNS server	172.25.250.254

Set the server's Ethernet interface to use the updated network settings displayed in the table above.

- Ensure that the host name of **serverb** is statically set to **server-review4.lab4.example.com**.
- On **serverb**, set **client-review4** as the canonical host name for the IPv4 address **172.25.250.10** of the host **servera.lab.example.com**.
- Configure the additional IPv4 address **172.25.250.211** with the netmask **255.255.255.0** on the same interface of **serverb** that has the existing static network settings. Do not remove the existing IPv4 address. Make sure that **serverb** responds to all addresses when the connection you statically configured on its interface is active.
- On **serverb**, restore the original network settings by activating the original network connection and deactivating the **static** network connection you created manually.

Evaluation

On **workstation**, run the **lab rhcsa-rh124-review4 grade** command to confirm success of this exercise.

```
[student@workstation ~]$ lab rhcsa-rh124-review4 grade
```

Finish

On **workstation**, run **lab rhcsa-rh124-review4 finish** to complete the comprehensive review. This script deletes the files and directories created during the start of the comprehensive review and ensures that the environment on **serverb** is clean.

```
[student@workstation ~]$ lab rhcsa-rh124-review4 finish
```

This concludes the comprehensive review.

► Solution

Managing Networks

In this review, you will configure and test network connectivity.

Outcomes

You should be able to:

- Configure the network settings.
- Test network connectivity.
- Set a static host name for the system.
- Use locally resolvable canonical host names to connect to systems.

Before You Begin

Log in to **workstation** as **student** using **student** as the password.

On **workstation**, run **lab rhcsa-rh124-review4 start** to start the comprehensive review. This script creates the necessary files to set up the environment correctly.

```
[student@workstation ~]$ lab rhcsa-rh124-review4 start
```

Instructions

Accomplish the following tasks on **serverb** to complete the exercise.



Warning

It is a useful practice to make network changes from the server console, whether locally or through remote console access hardware. When using **ssh** to adjust networking settings, a mistaken command can hang or lock out your session. Network configuration corrections must then be made through the console.

In the web page that controls your lab environment, click the **OPEN CONSOLE** button for **serverb**. A tab will open in your browser with the console session for **serverb**. Log in as user **student** at the prompt.

- Determine the name of the Ethernet interface and its active connection profile on **serverb**.
- On **serverb**, create a new connection profile called **static** for the available Ethernet interface that statically sets network settings and does not use DHCP. Use the settings in the following table.

IPv4 address	172.25.250.111
Netmask	255.255.255.0

Gateway	172.25.250.254
DNS server	172.25.250.254

Set the server's Ethernet interface to use the updated network settings displayed in the table above.

- Ensure that the host name of **serverb** is statically set to **server-review4.lab4.example.com**.
- On **serverb**, set **client-review4** as the canonical host name for the IPv4 address **172.25.250.10** of the host **servera.lab.example.com**.
- Configure the additional IPv4 address **172.25.250.211** with the netmask **255.255.255.0** on the same interface of **serverb** that has the existing static network settings. Do not remove the existing IPv4 address. Make sure that **serverb** responds to all addresses when the connection you statically configured on its interface is active.
- On **serverb**, restore the original network settings by activating the original network connection and deactivating the **static** network connection you created manually.

1. Use the console to login as **student** to **serverb** locally.
 - 1.1. In the web page that controls your lab environment, click the **OPEN CONSOLE** button for **serverb**. A tab will open in your browser with the console session for **serverb**. Log in as user **student** at the prompt.
It is a useful practice to make network changes from the server console, whether locally or through remote console access hardware. When using **ssh** to adjust networking settings, a mistaken command can hang or lock out your session. Network configuration corrections must then be made through the console.
2. Determine the Ethernet interface name on **serverb** and the connection profile name it uses.
 - 2.1. In this example, **enX** is the Ethernet interface name. The connection profile name is **Wired connection 1**. Create the **static** connection profile for this interface.
 - 2.2. The network interface and initial connection profile names might differ on your **serverb**. Use the name shown by your system to replace the **enX** placeholder name in the steps of this solution.
3. On **serverb**, create a new connection profile called **static** for the available Ethernet interface. Set the network settings statically so that it does not use DHCP. Base the settings on the following table:

IPv4 address	172.25.250.111
Netmask	255.255.255.0
Gateway	172.25.250.254
DNS server	172.25.250.254

The **serverb** server's Ethernet interface should use the updated network settings as mentioned in the preceding table.

- 3.1. Use **nmcli** to create the **static** connection with the given network settings.

```
[student@serverb ~]$ sudo nmcli connection add con-name static type ethernet \
  ifname enX ipv4.addresses '172.25.250.111/24' ipv4.gateway '172.25.250.254' \
  ipv4.dns '172.25.250.254' ipv4.method manual
[sudo] password for student: student
Connection 'static' (ac8620e6-b77e-499f-9931-118b8b015807) successfully added.
```

- 3.2. Use the **nmcli** command to activate the new connection settings.

```
[student@serverb ~]$ sudo nmcli connection up static
```

4. Use the **hostnamectl** command to set the **serverb** host name to **server-review4.lab4.example.com**. Verify the new host name.

- 4.1.

```
[student@serverb ~]$ sudo hostnamectl set-hostname server-review4.lab4.example.com
[sudo] password for student: student
[student@serverb ~]$ hostname
server-review4.lab4.example.com
```

5. On **serverb**, edit the **/etc/hosts** file to set **client-review4** as the canonical host name for the IPv4 address **172.25.250.10** of the host **servera.lab.example.com**.

- 5.1. Edit the **/etc/hosts** file to add **client-review4** as a name for the **172.25.250.10** IPv4 address.

```
172.25.250.10 servera.lab.example.com servera client-review4
```

- 5.2. Use the **ping** command to verify that you can reach **172.25.250.10** using the canonical host name **client-review4**.

```
[student@serverb ~]$ ping -c2 client-review4
PING servera.lab.example.com (172.25.250.10) 56(84) bytes of data.
64 bytes from servera.lab.example.com (172.25.250.10): icmp_seq=1 ttl=64
time=0.259 ms
64 bytes from servera.lab.example.com (172.25.250.10): icmp_seq=2 ttl=64
time=0.391 ms

--- servera.lab.example.com ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 33ms
rtt min/avg/max/mdev = 0.259/0.325/0.391/0.066 ms
```

6. Modify the connection profile **static** to configure the additional IPv4 address **172.25.250.211** with the netmask **255.255.255.0** on the same **serverb** interface with the existing static settings. Do not remove the existing IPv4 address. Verify that **serverb** responds to all addresses when the modified connection profile is active.

- 6.1. Use the **nmcli** command to add the new IP address.

```
[student@serverb ~]$ sudo nmcli connection modify static \
+ipv4.addresses '172.25.250.211/24'
```

- 6.2. Use the **nmcli** command to activate the new IP address.

```
[student@serverb ~]$ sudo nmcli connection up static
...output omitted...
```

- 6.3. From **workstation**, use the **ping** command to verify that the IPv4 address **172.25.250.211** can be reached.

```
[student@workstation ~]$ ping -c2 172.25.250.211
PING 172.25.250.211 (172.25.250.211) 56(84) bytes of data.
64 bytes from 172.25.250.211: icmp_seq=1 ttl=64 time=0.246 ms
64 bytes from 172.25.250.211: icmp_seq=2 ttl=64 time=0.296 ms

--- 172.25.250.211 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 50ms
rtt min/avg/max/mdev = 0.246/0.271/0.296/0.025 ms
```

7. On **serverb**, restore the original settings by activating the original network connection.

- 7.1. Return to the console and use the **nmcli** command to activate the original network profile.

```
[student@serverb ~]$ sudo nmcli connection up "Wired connection 1"
...output omitted...
```

The original connection profile name might differ on your **serverb**. Replace the name shown in this solution with the name from your system. Find the name with **nmcli connection show**.

- 7.2. From **workstation**, open an SSH session to **serverb** as **student** to verify that the original network settings are successfully activated.

```
[student@workstation ~]$ ssh student@serverb
...output omitted...
[student@server-review4 ~]$
```

- 7.3. Log out of **serverb** and exit all but one terminal on **workstation**.

```
[student@server-review4 ~]$ exit
logout
Connection to serverb closed.
[student@workstation ~]$
```

Evaluation

On **workstation**, run the **lab rhcsa-rh124-review4 grade** command to confirm success of this exercise.

```
[student@workstation ~]$ lab rhcsa-rh124-review4 grade
```

Finish

On **workstation**, run **lab rhcsa-rh124-review4 finish** to complete the comprehensive review. This script deletes the files and directories created during the start of the comprehensive review and ensures that the environment on **serverb** is clean.

```
[student@workstation ~]$ lab rhcsa-rh124-review4 finish
```

This concludes the comprehensive review.

► Lab

Mounting Filesystems and Finding Files

In this review, you will mount a file system and locate files based on different criteria.

Outcomes

You should be able to:

- Mount an existing file system.
- Find files on the basis of the file name, permissions and size.

Before You Begin

Log in to **workstation** as **student** using **student** as the password.

On **workstation**, run **lab rhcsa-rh124-review5 start** to start the comprehensive review. This script creates the necessary file system, user accounts and group accounts.

```
[student@workstation ~]$ lab rhcsa-rh124-review5 start
```

Instructions

Accomplish the following tasks on **serverb** to complete the exercise.

- On **serverb**, a block device containing the **XFS** file system exists but is not yet mounted. Determine the block device and mount it on the **/review5-disk** directory. Create the **/review5-disk** directory, if necessary.
- On **serverb**, locate the file called **review5-path**. Create a file named **/review5-disk/review5-path.txt** that contains a single line consisting of the absolute path to the **review5** file.
- On **serverb**, locate all the files having **contractor1** and **contractor** as the owning user and group, respectively. The files must also have the octal permissions of **640**. Save the list of these files in **/review5-disk/review5-perms.txt**.
- On **serverb**, locate all files 100 bytes in size. Save the absolute paths of these files in **/review5-disk/review5-size.txt**.

Evaluation

On **workstation**, run the **lab rhcsa-rh124-review5 grade** command to confirm success of this exercise.

```
[student@workstation ~]$ lab rhcsa-rh124-review5 grade
```

Finish

On **workstation**, run **lab rhcsa-rh124-review5 finish** to complete the comprehensive review. This script deletes the file system, user accounts, and group accounts created during the start of the comprehensive review and ensures that the environment on **serverb** is clean.

```
[student@workstation ~]$ lab rhcsa-rh124-review5 finish
```

This concludes the comprehensive review.

► Solution

Mounting Filesystems and Finding Files

In this review, you will mount a file system and locate files based on different criteria.

Outcomes

You should be able to:

- Mount an existing file system.
- Find files on the basis of the file name, permissions and size.

Before You Begin

Log in to **workstation** as **student** using **student** as the password.

On **workstation**, run **lab rhcsa-rh124-review5 start** to start the comprehensive review. This script creates the necessary file system, user accounts and group accounts.

```
[student@workstation ~]$ lab rhcsa-rh124-review5 start
```

Instructions

Accomplish the following tasks on **serverb** to complete the exercise.

- On **serverb**, a block device containing the **XFS** file system exists but is not yet mounted. Determine the block device and mount it on the **/review5-disk** directory. Create the **/review5-disk** directory, if necessary.
- On **serverb**, locate the file called **review5-path**. Create a file named **/review5-disk/review5-path.txt** that contains a single line consisting of the absolute path to the **review5** file.
- On **serverb**, locate all the files having **contractor1** and **contractor** as the owning user and group, respectively. The files must also have the octal permissions of **640**. Save the list of these files in **/review5-disk/review5-perms.txt**.
- On **serverb**, locate all files 100 bytes in size. Save the absolute paths of these files in **/review5-disk/review5-size.txt**.

1. On **serverb**, mount the idle block device containing the **XFS** file system on the **/review5-disk** directory.
 - 1.1. From **workstation**, open an SSH session to **serverb** as **student**.

```
[student@workstation ~]$ ssh student@serverb
...output omitted...
[student@serverb ~]$
```

- 1.2. Use the **lsblk -fs** command to determine the idle block device containing the **XFS** file system.

```
[student@serverb ~]$ lsblk -fs
NAME   FSTYPE LABEL UUID
...output omitted...
vdb1    xfs      3d97c5ef-23e7-4c1c-a9be-d5c475b3d0d5
└─vdb
...output omitted...
```

From the preceding output, note that the **vdb1** block device contains the **XFS** file system, which is not mounted on any directory.

- 1.3. Use the **sudo mkdir** command to create the **/review5-disk** directory as the superuser. When the **sudo** command prompts you for a password, give the password **student**.

```
[student@serverb ~]$ sudo mkdir /review5-disk
[sudo] password for student: student
```

- 1.4. Use the **sudo mount** command to mount the **vdb1** block device on the **/review5-disk** directory as the superuser.

```
[student@serverb ~]$ sudo mount /dev/vdb1 /review5-disk
```

- 1.5. Verify that the **vdb1** block device is successfully mounted on the **/review5-disk** directory.

```
[student@serverb ~]$ df -Th
Filesystem      Type      Size  Used Avail Use% Mounted on
...output omitted...
/dev/vdb1        xfs      2.0G  47M  2.0G  3% /review5-disk
...output omitted...
```

2. On **serverb**, locate the file named **review5-path**. Record its absolute path in the **/review5-disk/review5-path.txt** text file.

- 2.1. Use the **find** command to locate the file called **review5-path**. Redirect all the errors of the **find** command to **/dev/null**. This redirection allows you to discard any error from the output of the **find** command.

```
[student@serverb ~]$ find / -iname review5-path 2>/dev/null
/var/tmp/review5-path
```

Note the absolute path to the **review5-path** file from the preceding output.

- 2.2. Create the **/review5-disk/review5-path.txt** text file. Record the absolute path to the **review5-path** file, as determined in the preceding step, in the **/review5-disk/review5-path.txt** text file. You may use the **sudo vim /review5-disk/review5-path.txt** command to create the text file. Type **:wq!** from the command mode in **vim** to save the changes and quit from the file. The following output shows the content of the **/review5-disk/review5-path.txt** text file.

```
/var/tmp/review5-path
```

3. On **serverb**, locate all files having **contractor1** and **contractor** as the owning user and group, respectively. The files must also have the octal permissions of **640**. Record the absolute paths to all of these files in the **/review5-disk/review5-perms.txt** text file.
- 3.1. Use the **-user**, **-group**, **-perm** options with the **find** command to locate all the files that have the owning user, owning group and octal permissions of **contractor1**, **contractor** and **640**, respectively. Redirect all the errors of the **find** command to **/dev/null**.

```
[student@serverb ~]$ find / -user contractor1 \
-group contractor \
-perm 640 2>/dev/null
/usr/share/review5-perms
```

Note the absolute path to the **review5-perms** file from the preceding output. The **/usr/share/review5-perms** file is the only one that meets the criteria of the preceding **find** command.

- 3.2. Create the **/review5-disk/review5-perms.txt** text file. Record the absolute path to the only file (**review5-perms**) that has the owning user, owning group and octal permissions of **contractor1**, **contractor** and **640**, respectively, as determined in the preceding step, in the **/review5-disk/review5-perms.txt** text file. You may use the **sudo vim /review5-disk/review5-perms.txt** command to create the text file. Type **:wq!** from the command mode in **vim** to save the changes and quit from the file. The following output shows the content of the **/review5-disk/review5-perms.txt** text file.

```
/usr/share/review5-perms
```

4. On **serverb**, locate all the files of 100 bytes in size. Record the absolute paths to all of these files in the **/review5-disk/review5-size.txt**.
- 4.1. Use the **-size** option with the **find** command to locate all the files that are of 100 bytes in size. Redirect all the errors of the **find** command to **/dev/null**.

```
[student@serverb ~]$ find / -size 100c 2>/dev/null
/dev/disk
/run/initramfs
/etc/lvm
/etc/audit
/etc/sos.conf
/usr/lib/python3.6/site-packages/dnf/conf
/usr/lib/python3.6/site-packages/ptyprocess
/usr/share/licenses/ethtool/LICENSE
/usr/share/doc/libuser
/usr/share/doc/python3-cryptography/docs/x509
/usr/share/doc/python3-jinja2/ext
/usr/share/doc/plymouth/AUTHORS
/usr/share/vim/vim80/macros/maze/main.aap
/usr/libexec/plymouth
/opt/review5-size
```

The preceding output may vary in your system depending on the number of files of 100 bytes in size in your system. Note the absolute paths to all the files from the preceding output.

- 4.2. Create the **/review5-disk/review5-size.txt** text file. Record the absolute paths to all the files of 100 bytes in size, as determined in the preceding step, in the **/review5-disk/review5-size.txt** text file. You may use the **sudo vim /review5-disk/review5-size.txt** command to create the text file. Type **:wq!** from the command mode in **vim** to save the changes and quit from the file. The **/review5-disk/review5-size.txt** text file should contain the absolute path to the **review5-size** file among other paths.

```
...output omitted...
/opt/review5-size
...output omitted...
```

- 4.3. Log out of **serverb**.

```
[student@serverb ~]$ exit
logout
Connection to serverb closed.
[student@workstation ~]$
```

Evaluation

On **workstation**, run the **lab rhcsa-rh124-review5 grade** command to confirm success of this exercise.

```
[student@workstation ~]$ lab rhcsa-rh124-review5 grade
```

Finish

On **workstation**, run **lab rhcsa-rh124-review5 finish** to complete the comprehensive review. This script deletes the file system, user accounts, and group accounts created during the start of the comprehensive review and ensures that the environment on **serverb** is clean.

```
[student@workstation ~]$ lab rhcsa-rh124-review5 finish
```

This concludes the comprehensive review.

