**Red Hat System Administration I 8.2**

**Chapter 1: Getting Started with Red Hat Enterprise Linux**

**Index:**

* What is Linux?
* Quiz: Getting Started with Red Hat Enterprise Linux
* Summary

**Abstract:**

* 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

**Quiz: Getting Started with Red Hat Enterprise Linux**

1. **Which two of the following are benefits of open-source software for the user? (Choose two.)**
   1. Code can survive the loss of the original developer or distributor.
   2. Sensitive portions of code are protected and only available to the original developer.
   3. You can learn from real-world code and develop more effective applications.
   4. 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 its products for the future and interacts with the community? (Choose two.)**
   1. Sponsor and integrate open-source projects into the community-driven Fedora project.
   2. Develop specific integration tools only available in Red Hat distributions.
   3. Participate in upstream projects.
   4. Repackage and re-license community products.
3. **Which two statements describe the benefits of Linux? (Choose two.)**
   1. Linux is developed entirely by volunteers making it a low-cost operating system.
   2. Linux is modular and can be configured as a full graphical desktop or a small appliance.
   3. Linux is locked in a known state for a minimum of one year for each release, making it easier to develop custom software.
   4. Linux includes a powerful and scriptable command-line interface, enabling easier automation and provisioning.

**Chapter 01: 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, supporting user programs and libraries.
* Red Hat supports and contributes code to open-source projects and sponsors. It integrates project software into community-driven distributions, stabilizing 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**

**Index:**

* Accessing the Command Line
* Quiz: Accessing the Command Line
* Accessing the Command Line Using the Desktop
* Guided Exercise: Accessing the Command Line Using the Desktop
* Executing Commands Using the Bash Shell
* Quiz: Executing Commands Using the Bash Shell
* Lab: Accessing the Command Line
* Summary

**Abstract:**

* 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

**Chapter 2: Quiz: Accessing the Command Line**

1. **Which term describes the interpreter that executes commands typed as strings?**
   1. Command
   2. Console
   3. Shell
   4. Terminal
2. **Which term describes the visual cue that indicates an interactive shell is waiting for the user to type a command?**
   1. Argument
   2. Command
   3. Option
   4. Prompt
3. **Which term describes the name of a program to run?**
   1. Argument
   2. Command
   3. Option
   4. Prompt
4. **Which term describes the part of the command line that adjusts the behavior of a command?**
   1. Argument
   2. Command
   3. Option
   4. Prompt
5. **Which term describes the part of the command line that specifies the target that the command should operate on?**
   1. Argument
   2. Command
   3. Option
   4. Prompt
6. **Which term describes the hardware display and keyboard used to interact with a system?**
   1. Physical Console
   2. Virtual Console
   3. Shell
   4. Terminal
7. **Which term describes one of the multiple logical consoles that can each support an independent login session?**
   1. Physical Console
   2. Virtual Console
   3. Shell
   4. Terminal
8. **Which term describes an interface that provides a display for output and a keyboard for input to a shell session?**
   1. Console
   2. Virtual Console
   3. Shell
   4. Terminal

**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.

**Instructions:**

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

1. **Log in to the workstation as a student using the student as the password.**
   1. click the student user account on the GNOME login screen on the workstation. Enter student when prompted for the password.
   2. Click Sign In.
2. **Change the password for the student from student to 55TurnK3y.**
   1. **Important:** The finish script resets the password for the student user to the student. The script must be executed at the end of the exercise.
   2. The simplest approach is to open a Terminal window and use the passwd command at the shell prompt. In the virtual learning environment with a visual keyboard, press the Super key twice to enter Activities overview. Type terminal and then press Enter to start the Terminal.
   3. In the terminal window that displays, type passwd at the shell prompt. Change the student password from student to 55TurnK3y.

1. [student@workstation ~]$ passwd

2. Changing password for user student.

3. Current password: student

4. New password: 55TurnK3y

5. Retype new password: 55TurnK3y

6. passd: all authentication tokens updated successfully.

1. **Log out and log back in as student using 55TurnK3y as the password to verify the changed password.**
   1. Click the system menu in the upper-right corner.
   2. Select Student User → Log Out.
   3. Click Log Out in the confirmation dialog box that displays.
   4. At the GNOME login screen, click the student user account. Enter 55TurnK3y when prompted for the password.
   5. Click Sign In.
2. **Lock the screen.**
   1. From the system menu in the upper-right corner, press the lock screen button at the bottom of the menu.
3. Unlock the screen.
   1. Press Enter to lift the lock screen curtain.
   2. In the Password field, enter 55TurnK3y as the password.
   3. Click Unlock.
4. **Determine how to shut down the workstation from the graphical interface but cancel the operation without shutting down the system.**
   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.
   2. Click Cancel in the dialog box that displays.
5. **Finish**
   1. On workstation, run the lab cli-desktop finish script to complete this exercise.

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

* 1. 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 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 ~]$ pasTab+Tab**

passwd paste pasuspender

**[user@host ~]$ passTab**

**[user@host ~]$ passwd**

Changing password for user user.

Current password:

|  |  |
| --- | --- |
|  | Press **Tab** twice. |
|  | 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/pasTab**

**[user@host ~]$ ls /etc/passwdTab**

passwd passwd-

|  |  |
| --- | --- |
|  | 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 --Tab+Tab**

--base-dir --groups --no-log-init --shell

--comment --help --non-unique --skel

--create-home --home-dir --no-user-group --system

--defaults --inactive --password --uid

--expiredate --key --root --user-group

--gid --no-create-home --selinux-user

**[root@host ~]# useradd --**

|  |  |
| --- | --- |
|  | 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 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, protraying 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.

**Table 2.1. 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

**Chapter 2: Quiz Executing Commands Using the Bash Shell**

1. **Which Bash shortcut or command jumps to the beginning of the previous word on the command line?**
   1. Pressing Ctrl+LeftArrow
   2. Pressing Ctrl+K
   3. Pressing Ctrl+A
   4. !string
   5. !number
2. **Which Bash shortcut or command separates commands on the same line?**
   1. Pressing Tab
   2. history
   3. ;
   4. !string
   5. Pressing Esc+.
3. **Which Bash shortcut or command is used to clear characters from the cursor to the end of the command line?**
   1. Pressing Ctrl+LeftArrow
   2. Pressing Ctrl+K
   3. Pressing Ctrl+A
   4. ;
   5. Pressing Esc+.
4. **Which Bash shortcut or command is used to re-execute a recent command by matching the command name?**
   1. Pressing Tab
   2. !number
   3. !string
   4. history
   5. Pressing Esc+.
5. **Which Bash shortcut or command is used to complete commands, file names, and options?**
   1. ;
   2. !number
   3. history
   4. Pressing Tab
   5. Pressing Esc+.
6. **Which Bash shortcut or command re-executes a specific command in the history list?**
   1. Pressing Tab
   2. !number
   3. !string
   4. history
   5. Pressing Esc+.
7. **Which Bash shortcut or command jumps to the beginning of the command line?**
   1. !number
   2. !string
   3. Pressing Ctrl+LeftArrow
   4. Pressing Ctrl+K
   5. Pressing Ctrl+A
8. **Which Bash shortcut or command displays the list of previous commands?**
   1. Pressing Tab
   2. !string
   3. !number
   4. history
   5. Pressing Esc+.
9. **Which Bash shortcut or command copies the last argument of previous commands?**
   1. Pressing Ctrl+K
   2. Pressing Ctrl+A
   3. !number
   4. Pressing Esc+.

|  |  |  |  |
| --- | --- | --- | --- |
|  | Franklin VM: | Standard User Account: | The Student's Root Account: |
| Username | kiosk | student | root |
| Password | redhat | student | redhat |

[kiosk@foundation0 ~] $ rht-vmctl start all

Starting bastion.

Starting workstation.

Starting utility.

Starting servera.

Starting serverb.

[ kiosk@foundation0 ~] $ rht-vmview view workstation

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

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

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