A Practical Guide to Fedora™ and Red Hat® Enterprise Linux®

SEVENTH EDITION

Lab Manual
Instructor Version (Includes Answers)

Release 1.0

Mark G. Sobell

PRENTICE HALL

Upper Saddle River, NJ • Boston • Indianapolis • San Francisco
New York • Toronto • Montreal • London • Munich • Paris • Madrid
Capetown • Sydney • Tokyo • Singapore • Mexico City

ISBN-13 (PDF): 978-0-13-347786-3
ISBN-10 (PDF): 0-13-347786-X

Second release, March 2014

Please send comments and corrections to the author at mgs@sobell.com.

Contents

How to Use This Lab Manual

This Lab Manual is designed for use with Mark Sobell's *A Practical Guide to Fedora and Red Hat Enterprise Linux, Seventh Edition*. Each of the labs includes page references to the Sobell book.

This Lab Manual has students work on a system on which they install Fedora 19 (the *student system*). The labs assume the student is working in a virtual environment using VMware Player or on a stand-alone physical computer.

Hardware and Software Requirements

- **Student system**—The physical or virtual student system must have at least 1 gigabyte of RAM and 25 gigabytes of free disk space. When you use the ISO image file (and not when you use the DVD) you need an additional 4 gigabytes of free space. On a shared system, the disk space must be available on an external drive. Lab 1 and Lab 2 guide the student through setting up this system.

The student system and the classroom server

tip These labs refer to the system the student is working on as the **local system** or the **student system**. They refer to the classroom server system as the **classroom server** or **class**.

- **Classroom server**—The physical or virtual classroom server system must have at least 768 megabytes of RAM and 5 gigabytes of free disk space. This system can be installed by the student, provided by the instructor, and in some cases shared by the class. The classroom server is not required until Lab 18. Lab 46, which is available only to the instructor, explains how to set up the classroom server.

  Lab 18, step 1 on page 73 of this Lab Manual adds the name (**class**) and IP address of the classroom server to the **/etc/hosts** file of the student systems for use in all subsequent labs that make use of the classroom server. All students must follow these instructions to be able to follow the labs that refer to the classroom server as **class**.

- **Fedora 19 install DVD**—You can obtain this DVD from the back of the hardcopy version of the Sobell text. Alternately, you can download and use a copy of the Fedora 19 installation (ISO) image file; see Sobell, page 46 for more information. The DVD or ISO image file must be available on the student system. If you wish to use a version of Fedora 19 other than the 32-bit Intel version, you will need to download the ISO image file.

- **Nonprivileged users**—The primary user, which is created when you install the system, is named Sammy Student and has the username **student**. For simplicity, all nonprivileged users have the same password: **fit714tree**. Normally, when you set up a Linux system, each user will have her own password. See Sobell, page 136 for information on selecting secure passwords.

Because these labs build on preceding labs, it is important that you set up users as described in the labs.

- Lab 7 (page 30) adds **ben**, **max**, and the **linux** group.
- Lab 13 (page 56) adds **casey** and the **staff** group.
- Lab 14 (page 60) adds **rose**.

- **Student's initials**—Several labs ask you to write files to the classroom server. To keep students from overwriting other students' files, you are asked to append your initials (***XXX***) to the names of these files. Before you start working with this Lab Manual, make sure that each student uses initials that are unique among the students that will be working with the classroom server.
- **Privileged user** (**root**)—The privileged user with the username **root**, also referred to as Superuser, has the password **bird482dog**. You also set up this user when you install the system.

If the student system does not have Internet access

tip A few labs require access to software packages that are not included on the Fedora 19 install DVD. These labs explain how to download and install these packages using an Internet connection. For student systems without Internet connectivity, these packages must be downloaded in advance and made available to the student system. The packages that are required but not included on the Fedora 19 install DVD are:

- **autofs**
- **openldap-servers**

You can download these packages from download.fedoraproject.org, which redirects to a mirror site. Navigate to the **releases/19/Fedora/i386/os/Packages** directory (replace i386 with x86_64 if you are using a 64-bit system). Download these packages to the student system or to removable media.

Additionally, Lab 32 requires the source code tarball for the which utility (**which-2.20.tar.gz**). You can download this file from ftp://ftp.gnu.org/gnu/which and make it available on the student system.

Answers

Answers appear in the Instructor Version of this Lab Manual; answers do not appear in the Student Version.

```
$ echo 'example of an answer' # Appears only in Instructor Version
```

Please send any corrections, additions, or comments to the author at mgs@sobell.com. Mention the version of this Lab Manual as shown on the title page.

Lab 1: Setting Up the Student System (10–15 minutes)

The first part of this lab describes how to set up a virtual student system (VM or virtual machine) using VMware Player (Option 1; next). The second part describes how to set up a stand-alone physical system (Option 2; page 9 of this Lab Manual). Both options explain how to set up the system so that you can install Fedora from the Install Image DVD or the DVD ISO image file and end with the system paused with the Install Image Boot menu (Sobell, Figure 3-4 on page 61) displayed on the screen. The installation steps are covered in Lab 2 (page 11).

The student system and the classroom server

tip These labs refer to the system the student is working on as the **local system** or the **student system**. They refer to the classroom server system as the **classroom server** or **class**.

Objectives

In this lab you will set up a VMware instance (Option 1) or stand-alone system (Option 2) on which you will subsequently install Fedora.

Setup

- VMware Player version 6 or a stand-alone physical system
- Fedora 19 Install Image DVD or, on a VM only, a DVD ISO image file on the host system
- 25 gigabytes of free disk space on either
  - An external USB hard disk *or*
  - The VM or stand-alone system (if no one else will be using it)

Option 1: Setting Up a Student System Using VMware Player

Procedure

1. Create a new VM (virtual machine) for the Fedora installation. See "VMware Player: Installing Fedora on VMware" on page 671 of Sobell for detailed instructions. Simplified instructions follow.

   a. Launch VMware Player.

   b. Click **Create a New Virtual Machine**.

   c. Click the radio button labeled **I will install the operating system later** and then click **Next**.

   d. Click the radio button labeled **Linux** and select **Fedora** from the drop-down list labeled **Version**. Click **Next**.

e. The default name of the VM is **Fedora**. Use the text box labeled **Name** to change the name of the VM so that it is unique within your class. Use the format *Fedora.**XXX*** where ***XXX*** are your initials.

   The text box labeled **Location** allows you to specify where you want to store the VM image (the files that define the VM). Select one of the following locations.

   - If several people share the local system, you can store the VM image on an external USB hard disk. Plug the disk into the local (student) system; the operating system will mount it. Once it is mounted, specify the mount point in the text box labeled **Location** (you can click **Browse** to find it). Make sure the location includes the name you just specified at the end of the pathname (e.g., **/vmw/Fedora.STG**).
   - If you are the only user on the local system, you can accept the default location as long as there is enough space on the host hard disk to hold the VM.
   - Click **Next**.

f. Accept the default maximum disk size of 20 gigabytes (GB). Click the radio button labeled **Store virtual disk as a single file**; click **Next**.

g. Review the information in the next window and Click **Finish**. If a message about installing VMware Tools appears, click **Close** to return to the VMware Player window. Do not install VMware Tools.

2. Get VMware Player ready to boot to install Fedora 19.

   a. To use VMware Player to access the Fedora 19 install image, click the name of the Fedora image you just created on the left side of the window and click **Edit Virtual Machine settings** near the bottom of the window on the right side; VMware Player displays the Hardware tab of the Virtual Machine Settings window.

   b. Click **CD/DVD (IDE)** on the left side of the Virtual Machine Settings window. Make sure the check box labeled **Connect at power on** has a tick in it. If you are using physical install media (a DVD), click the radio button labeled **Use a physical drive** and select the mount point for the DVD from the drop-down list labeled **Device**. If you are using a DVD ISO image, click **Use ISO image** and enter the pathname of the image file (you can click **Browse** to find it). Click **Save**.

Do not take this step unless the installation has previously failed

tip This step causes installation on most machines to fail but enables installation on a few older machines to work. The following steps assume the Virtual Machine Settings window selected in step 2a is still open and displaying the CD/DVD tab.

- Click **Advanced** to display the CD/DVD Advanced Settings window and place a tick in the check box labeled **Use legacy emulation on physical devices**.
- Click **Close** to return to the Virtual Machine Settings window.
- Click **Save** to save the settings.

c. To start the installation, click the name of the Fedora image you just created on the left side of the window and click **Play virtual machine** near the bottom of the window on the right side.

d. Press CONTROL-G or click while the mouse pointer is over the VM window to direct input to the VM. (Press this key or click to make selections during the installation so that what you type goes to the VM and not to the host desktop.)

e. When the VM displays the Install Image Boot menu (Sobell, Figure 3-4 on page 61), click on the window and then press the SPACE bar to pause the countdown. Ignore the messages that appear in boxes at the lower-right of the window; they will disappear after a few seconds.

f. Press CONTROL-ALT to return the cursor to the host desktop.

3. Continue the installation with Lab 2 (page 11).

The next installation steps as described in Lab 2 are the same whether you are installing to a stand-alone physical system or a VM.

Option 2: Setting Up a Student System on a Stand-alone Physical System

Although these labs focus on a student system that is installed on a VM, if you have the resources there is no reason you cannot set up the student system on a stand-alone physical system. Most of these labs will work exactly the same way with either setup; differences are noted.

Warnings

caution

- There is no information provided in these labs for a configuring dual-boot system. See Sobell, page 84 for more information.
- You are about to remove all information from the hard disk of the local machine. *You will lose all data on the machine.* Back up any data you want to save.

Procedure

1. Get the stand-alone physical system ready to boot to install Fedora 19.
   a. Place the installation DVD in the DVD drive.
   b. Boot the machine.
   c. When the system displays the Install Image Boot menu (Sobell, Figure 3-4 on page 61), press the SPACE bar to pause the countdown.
2. Continue the installation with Lab 2 on the next page.

Deliverables

A VM or stand-alone physical system that is paused while displaying the Install Image Boot menu (Sobell, Figure 3-4 on page 61).

Lab 2: Installing Fedora (30–50 minutes)

Objectives

In this lab you will work through the Anaconda installation, including customizing partitions, to install Fedora 19. This lab continues where Lab 1 left off.

Setup

- A VM or physical system that is paused while displaying the Install Image Boot menu (Sobell, Figure 3-4 on page 61).

Reading

Read the following sections:

- "Installing from an Install Image" on page 60 of Sobell
- "Manual/Custom Partitioning" on page 74 of Sobell

Procedure

These steps continue from Lab 1 and assume the Install Image Boot menu (Sobell, Figure 3-4 on page 61) is displayed on the screen and paused. If you are using a VM, click on the VM window so the host directs your keystrokes to the VM. Press CONTROL-ALT when you want to once again direct your keystrokes to the host.

1. Begin the installation.

   a. Use the ARROW keys to highlight **Test this media & install Fedora 19** and press RETURN to begin the installation. If you have already tested the media you do not need to do so again: Highlight **Install Fedora 19** and press RETURN.

      Ignore the **Press the <ENTER> key to begin the installation process** message; the installation will continue after a moment without intervention.

      Testing the install medium takes a few minutes. After Anaconda completes the test it displays the Welcome to Fedora 19 screen.

   b. Highlight the language you want to use during the installation; click **Continue**. After a moment Anaconda displays the Installation Summary screen (Sobell, Figure 3-6 on page 64).

   c. Click **Network Configuration** and use the text box labeled **Hostname** to change the hostname to **linux.example.com**. Click **Done** at the upper-left corner of the screen to redisplay the Installation Summary screen.

   d. Scroll down and click **Installation Destination** to display the Installation Destination screen (Sobell, Figure 3-9 on page 73). The frame labeled **Local Standard Disks** will show a single 20-gigabyte (approximately) disk with a tick in a circle on it. Click **Done** at the upper-left corner of the screen; Anaconda displays the Installation Options window.

2. Set up the initial disk partitioning.

   a. In the Installation Options window (Sobell, Figure 3-10 on page 74), click the radio button labeled **I want to review/modify my disk partitions before continuing**, make sure LVM is selected from the drop-down list labeled **Partition scheme**, and click **Continue**. Anaconda displays the Manual Partitioning window (Sobell, Figure 3-11 on page 75). Do not be concerned if the whole window is not displayed, the next step will remedy this situation.

   b. Create a default set of partitions by clicking **Click here to create them automatically.** The Manual Partitioning screen will look similar to that shown in Figure 3-11 on page 75 of Sobell, but the sizes of the partitions will be different.

3. Modify the partitioning as explained in "Manual/Custom Partitioning" on page 74 of Sobell. See also Sobell, page 40 for information on the swap partition and Sobell, page 44 for information on LVM.

   a. **Change the size of the swap partition—**Highlight the swap partition on the left side of the screen; the right side of the screen shows information about the LV (logical volume) named **fedora_linux-swap**. In the text box labeled **Desired Capacity**, replace the existing value with **1 GB** and click **Update Settings**; the left side of the screen reflects the change. The box labeled **Available Space** at the lower-left corner of the screen shows approximately 1 gigabyte (GB).

   b. **Change the size of the / (root) partition—**Highlight the **/** (root) partition on the left side of the screen; the right side of the screen shows information about the LV named **fedora_linux-root**. In the text box labeled **Desired Capacity**, replace the existing value with **15 GB** and click **Update Settings**; the left side of the screen reflects the change. The box labeled **Available Space** at the lower-left corner of the screen shows approximately 4 gigabytes.

   c. **Add a new partition—**Click the plus sign (**+**) just above the box labeled **Available Space**; Anaconda displays the Add a New Mount Point window (Sobell, Figure 3-13 on page 76). Select **/home** from the drop-down list labeled **Mount Point** and enter **2 GB** in the text box labeled **Desired Capacity**. Click **Add mount point**. Anaconda redisplays the Manual Partitioning screen showing the new **/home** partition. The box labeled **Available Space** at the lower-left corner of the screen shows approximately 2 gigabytes.

   d. **Accept the changes—**Click **Done** at the upper-left corner of the Manual Partitioning screen; Anaconda displays the Summary of Changes window. Click **Accept Changes**; Anaconda redisplays the Installation Summary window.

4. Begin the installation.

   a. Click **Begin Installation**; Anaconda displays the User Settings screen.

   b. See "User Settings" on page 68 of Sobell for instructions on how to add **bird482dog** as the **root** password (click **Root Password**) and how to add a user with a full name of **Sammy Student**, the username **student**, and the password **fit714tree** (click **User Creation**). Do *not* make **student** an administrator. Do require a password. You will have to click **Done** twice in each window because you are specifying a weak password. See Sobell, page 136 for information on choosing a secure password.

   c. Return to the Configuration screen and wait for the installation to complete. Installing Fedora takes a long time; watch the progress bar to see how much of the installation is complete.

5. Finish up.

   a. When Anaconda displays the message near the bottom of the window saying it has successfully installed Fedora and asks you to reboot the system, click **Reboot** and, if you are using a DVD, remove it.

If you are using VMware Player and a DVD and the installation fails at this point

tip If you are installing Fedora from a DVD (and not an ISO image file) using VMware Player and the installation fails at this point, start over from the beginning of Lab 1 (page 7). This time, follow the instructions in the tip on page 9 of this Lab Manual. A failed installation leaves files for the virtual machine on the disk. Either remove this virtual machine before trying to install it again, or give the new virtual machine a different name.

   b. The system reboots and, after a few moments, displays the Login screen (Sobell, Figure 4-1 on page 91).

Deliverables

A newly installed Fedora 19 system booted and waiting for Sammy Student (username **student**) to log in.

Lab 3: Logging In, Running gnome-initial-setup, and Running GUI Applications (10–15 minutes)

Objectives

When a user logs in for the first time, Fedora runs gnome-initial-setup. In this lab you will log in, work through the questions gnome-initial-setup asks, run GUI applications including a terminal emulator, and explore the system using the GUI.

These labs use the GNOME 3 Standard desktop

tip These labs explain how to run commands from a GNOME 3 Standard desktop. If you are running RHEL (Red Hat Enterprise Linux) or if you want to run commands from a GNOME 3 Classic desktop, refer to "The GNOME 3 Standard and Classic Desktops" on page 91 of Sobell.

Two ways to update the installed software

tip When you install Fedora from a DVD or DVD ISO image, there is a backlog of updates that you can apply to bring the installed software up to date. If the local system is connected to the Internet, it will periodically remind you that software updates are available; you can click this notice to update the software. Alternately, you can use yum to update the system as explained in Lab 8, step 1 on page 34 of this Lab Manual. If the system is not connected to the Internet, you cannot easily update the software. You can, however, install new software; see Lab 45 (page 151).

Setup

- A Fedora 19 installation waiting for Sammy Student (username **student**) to log in for the first time. You must know the **root** password.

Reading

Read the following sections:

- "gnome-initial-setup: Setting Up a User" on page 68 of Sobell
- "Logging In on the System" on page 90 of Sobell
- "The GNOME 3 Standard Desktop (Fedora)" on page 92 of Sobell
- "Working with the Desktop" on page 97 of Sobell
- "The Settings Window" on page 107 of Sobell

Procedure

Part I: Logging In

1. With the GNOME Login screen displayed (Sobell, Figure 4-1 on page 91), click **Sammy Student** and enter **fit714tree** in the text box labeled **Password**. Press RETURN or click **Sign In**.

Part II: Running gnome-initial-setup

2. The gnome-initial-setup utility customizes a user account by asking a few questions.

   a. In the Welcome screen click to put a tick next to the language you want to work in; click **Next**.

   b. In the Input Sources screen click **Next**. See "Input Sources" on page 68 of Sobell for more information.

   c. In the Online Accounts screen click **Next**.

   d. In the Thank You screen click **Start using GNOME 3**.

   e. Watch the short presentation on GNOME 3 and then close the GNOME Help window by clicking the **X** at the upper-right corner of the window.

Part III: Running GUI Applications

3. When you log in on a GNOME desktop you are using a GUI (graphical user interface). There are several ways to run a GUI application. These labs use a simple, consistent technique for running all GUI programs.

   a. Press the SUPER (Windows) key to display the Search text box as shown in Figure 4-2 on page 93 of Sobell.

   b. Type the name of the program you want to run. You only have to type enough of the name to uniquely identify the program. When a single application icon appears below the Search text box, you have uniquely identified the application.

You can type the application name or window name in the Search text box

tip You can type the name of the (graphical) application (e.g., **gnome-terminal** or **gnome-disks**) or the name of the window the application displays (e.g., **Terminal** or **Disks**) in the Search text box before pressing RETURN to open that application in the active workspace. If the Search text box is not visible, press the SUPER (Windows) key to display it.

   c. Press RETURN; GNOME opens the application window.

   d. To close a window, select **File⇨Close** from the application menubar (Sobell, Figure 4-8 on page 100), select Quit from the Application menu (Sobell, Figure 4-7 on page 99), or if nothing else works, click the X at the upper-right corner of the window.

4. Experiment by opening and then closing the gnome-disks (Disks), yelp (Help), and gedit (Gedit) GUI applications.

Part IV: Opening a Terminal Emulator Window

A terminal emulator is a GUI application that displays a CLI (command-line interface). *Terminal* is the name of the GNOME terminal emulator application. Open a window running Terminal by pressing the SUPER (Windows) key, typing **terminal**, and then pressing RETURN. Figure 4-23 on page 120 of Sobell shows a Terminal (emulator) window.

Part V: Explore the System Using the GUI

5. This step explores additional installation settings. See Part III on the prevous page for instructions on running the GUI applications described in this section.

   You can close each of the applications by selecting **Quit** from the Application menu (click the name of the window [e.g., Settings] at the top of the screen) or by clicking the **X** at the upper-right corner of the window.

   a. Display disk information by running the gnome-disks application (Sobell, page 78). Look at the items under Other devices (on the left); verify the size of the swap, **/home**, and **/** (root) partitions.

   b. Use the Settings window (Sobell, Figure 4-12 on page 107) to change the date and time settings, including the time zone. Click **Sammy Student** at the upper-right corner of the GNOME desktop and select **Settings**. Alternately, you can run **settings** as a GUI application. Click **Date & Time** at the bottom of the window (Sobell, page 110). Click **Unlock** at the upper-right corner of the window, enter the **root** password in the Authentication Required window, and click **Authenticate**. Select the local time zone as explained in the text. Adjust the time and date as necessary. If you have an Internet connection, click the slider labeled **Network Time** to turn on NTP (Network Time Protocol). You can either close this window or you can click the back arrow at the upper-left corner of the window to return to the Settings window.

   c. Use the Settings window to change the desktop background (Sobell, page 109). Open the Settings window as explained in step b. Click **Background** at the top of the window and then click the picture of the desktop in the Background window; GNOME displays several backgrounds you can choose from. Click Colors or Wallpapers at the top of the window, click a background, and click **Select** to change the background that GNOME displays.

   d. Verify network connectivity using the Settings window (see step b). Click **Network** in the Hardware section of the Settings window to display the Network window (Sobell, page 639). Answer the following questions. You will have to click the gear at the lower-right corner of the window and select among different tabs to answer some of the questions. Do not change any settings.

      - What is the IP address of the local system?
      - What is the IP address of the DNS server the local system is using?

- What is the name of the network connection (Identity tab)?
- Is the local system using DHCP to set itself up (IPv4 tab)?
- Is IPv6 turned on (IPv6 tab)?

Answers will vary from system to system.

Stopping GNOME from blanking the screen

tip As installed, GNOME blanks the screen after five minutes of inactivity. When it blanks the screen, you must enter your password to bring the screen back to life. This setup can be annoying when you repeatedly leave your work for a while and then return: Each time you return you must sign in again. You can adjust how often GNOME blanks the screen or you can turn this feature off altogether.

Open the Settings window (click **Sammy Student** at the upper-right corner of the screen and select **Settings**. From the Settings window, click **Power** in the Hardware section. Adjust the frequency with which GNOME blanks the screen by selecting a value from the drop-down list labeled **Blank Screen**. The bottom value is **Never**. Click the **X** at the upper-right corner of the Settings window to close it. GNOME saves the change automatically.

Deliverables

A Fedora 19 desktop running GNOME 3 with the user named **student** logged in. Also, practice using the GUI to examine and configure a user account.

Lab 4: Getting Started Using bash (20 minutes)

Objectives

In this lab you will determine which shell you are running, correct mistakes on the command line, and use filename generation. You will also learn about help the system provides in the form of man pages, the **––help** option, and the bash **help** command.

Use yelp for help with GUI commands/windows

tip Open the yelp (Help) GUI application for help with GUI commands/windows. See “Running GUI Applications” on page 15 of this Lab Manual for instructions opening this and other windows.

Setup

- A Fedora 19 installation that includes a nonprivileged user named **student**.

Reading

Read the following sections:

- “Conventions Used in This Book” on page 26 of Sobell
- “Working from the Command Line” on page 119 of Sobell
- “Special Characters” on page 142 of Sobell
- “Filename Generation/Pathname Expansion” on page 165 of Sobell

Procedure

When you work from the command line, you are working with bash (the Bourne Again Shell). This shell is discussed in detail in upcoming labs.

Lab Manual and book use the same conventions

tip This Lab Manual uses the same conventions as the Sobell book. Before getting started, read “Conventions Used in This Book” on page 26 of Sobell to become familiar with the conventions used in this Lab Manual.

The script utility can provide a record of your work with the shell

tip Use the script utility (Sobell, page 257) to capture your work as you go through the steps of this and other labs that use the CLI. You will end up with a file that shows the work you have done. You can submit this file to your instructor or keep it for your notes.

1. Open a terminal emulator window as explained on page 15 of this Lab Manual.

2. Type some words on a command line without pressing RETURN. Now delete a character, a word, and the entire line as explained at "Correcting Mistakes" on page 122 of Sobell.

   The default erase key is BACKSPACE, the default key for deleting a word is CONTROL-W, and the default line kill key is CONTROL-U or CONTROL-X. See "Erasing a Character" on page 123 of Sobell, for instructions on resetting these values.

3. The echo command (Sobell, page 219) displays its arguments (the words that follow it on the command line). Give the following command to cause echo to display the name of the shell you are running (Sobell, page 122):

```
$ echo $0
```

   If you are running bash, the shell displays

```
$ echo $0
bash
```

   or

```
$ echo $0
-bash
```

   or

```
$ echo $0
/bin/bash
```

4. When the shell sees a special character (Sobell, page 142) on the command line, it might change that character or take some other action. As step 3 demonstrates, the shell interprets a word that starts with a dollar sign as a variable: The shell replaced **$0** with the value held by the **$0** variable, which in this case is the name of the shell.

   a. On the command line and in shell scripts, the hash sign (#) special character starts a comment. These labs use comments to give you more information about commands. Do not type the hash sign and the words that follow it on a command line (although it will have no effect if you do). Try giving this command (and pressing RETURN) and see what happens:

```
$ # you can type anything here
```

   b. Another way the shell changes special characters is by expanding them to match the names of files (filename generation; Sobell, page 165). For example, it expands an unquoted asterisk (*) to match any part of a filename. Try giving the following commands:

```
$ echo *          # lists the names of all files in the
                  # working directory
$ echo /bin/a*    # lists all files in the /bin directory
                  # that start with the letter a
$ echo /bin/*a    # lists all files in the /bin directory
                  # that end with the letter a
```

5. You can use man (Linux manual) pages to learn about Linux utilities and other aspects of a Linux system (Sobell, page 128). Display and read the man page on the man utility to find out more about this utility.

   Give the following command to display the man page covering the man utility:

   ```
   $ man man
   ```

   Press the SPACE bar to display the next screen of information and press **q** to quit using man.

6. Many utilities accept the **––help** option (Sobell, page 133). When called with this option, one of these utilities displays information about itself. Call the cat utility with the **––help** option.

```
$ cat --help
Usage: cat [OPTION]... [FILE]...
Concatenate FILE(s), or standard input, to standard output.

  -A, --show-all           equivalent to -vET
  -b, --number-nonblank    number nonempty output lines
  -e                       equivalent to -vE
  -E, --show-ends          display $ at end of each line
  -n, --number             number all output lines
  -s, --squeeze-blank      suppress repeated empty output lines
  -t                       equivalent to -vT
  -T, --show-tabs          display TAB characters as ^I
  -u                       (ignored)
  -v, --show-nonprinting   use ^ and M- notation, except for LFD and TAB
      --help     display this help and exit
      --version  output version information and exit

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

Examples:
  cat f - g  Output f's contents, then standard input, then g's contents.
  cat        Copy standard input to standard output.

Report cat bugs to bug-coreutils@gnu.org
GNU coreutils home page: <http://www.gnu.org/software/coreutils/>
General help using GNU software: <http://www.gnu.org/gethelp/>
For complete documentation, run: info coreutils 'cat invocation'
```

7. Many utilities display a help message that is so long it scrolls off the screen. In this case you can use a pipeline (discussed in Lab 15; Sobell, page 158) and a pager such as less (pages 129 and 134 of Sobell) to display one screen of information at a time. Call the cp utility, specifying the **––help** option and sending the output through a pipeline to less.

```
$ cp --help | less
Usage: cp [OPTION]... [-T] SOURCE DEST
  or:  cp [OPTION]... SOURCE... DIRECTORY
```

```
  or:  cp [OPTION]... -t DIRECTORY SOURCE...
Copy SOURCE to DEST, or multiple SOURCE(s) to DIRECTORY.

Mandatory arguments to long options are mandatory for short options too.
  -a, --archive                same as -dR --preserve=all
      --backup[=CONTROL]       make a backup of each existing destination
...
```

Press the SPACE bar to display the next screen of information and **q** to quit using less.

8. The bash **help** command (Sobell, page 134) displays information about bash features. Use the bash **help** command to display information about the echo builtin.

```
$ help echo
echo: echo [-neE] [arg ...]
    Write arguments to the standard output.

    Display the ARGs on the standard output followed by a newline.

    Options:
      -n        do not append a newline
      -e        enable interpretation of the following backslash escapes
      -E        explicitly suppress interpretation of backslash escapes
...
```

DELIVERABLES

This lab gives you practice correcting mistakes on the command line and getting assistance with Linux using man, the **––help** option, and the bash **help** command.

Lab 5: Introducing a Few Utilities (25 minutes)

Objectives

In this lab you will learn how to create a short file using basic vim editor commands and how to use a few common utilities. The utilities will be used in their simplest forms. Most of the utilities have many options and can be used in many ways. Use the **––help** option, refer to Sobell, Chapter 7, or display the man page for a utility to learn more about it.

Setup

- A Fedora 19 installation that includes a nonprivileged user named **student**. You must know the **root** password.

Reading

Read the following sections:

- "Tutorial: Using vim to Create and Edit a File" on page 262 of Sobell
- "ls: Displays Information About Files" on page 221 of Sobell
- "cp: Copies Files" on page 224 of Sobell
- "rm: Removes a File (Deletes a Link)" on page 222 of Sobell
- "less Is more: Display a Text File One Screen at a Time" on page 220 of Sobell
- "head: Displays the Beginning of a File" on page 235 of Sobell
- "tail: Displays the Last Part of a File" on page 241 of Sobell
- "hostname: Displays the System Name" on page 219 of Sobell
- "mv: Renames or Moves a File" on page 237 of Sobell
- "lpr: Sends Files to Printers" on page 235 of Sobell
- "grep: Searches for a Pattern in Files" on page 232 of Sobell
- "sort: Sorts and/or Merges Files" on page 239 of Sobell
- "file: Displays the Classification of a File" on page 229 of Sobell

Procedure

Most of the utilities this lab introduces work with files. Before you can use these utilities, you must have a file to work with. The first step in this lab explains the basics of how to use the vim editor to create a file. Lab 9 of this manual explains how to use vim to edit and correct mistakes in files.

1. Before you start using vim, install the **vim-enhanced** package. If you do not install this package, you will have to call vim as vi and it will lack useful features. The su and yum utilities are discussed in an upcoming lab.

   ```
   $ su -c 'yum -y install vim-enhanced'
   Password:
   ```

   Enter the **root** password in response to the **Password** prompt and then press RETURN.

2. The following instructions explain how to create a short file. You can correct a mistake on the line you are entering by using the correction keys explained in step 2 of Lab 4 (Lab Manual, page 19). If you notice a mistake on a previous line, leave it; you will learn how to correct these kinds of mistakes in a later lab.

   a. Give the following command to open the vim editor so it is editing the file named **practice**.

      ```
      $ vim practice
      ```

      The screen will look like Figure 7-1 on page 263 of Sobell.

      If bash displays a **command not found** error, see step 1 or give the preceding command again, replacing **vim** with **vi**.

   b. Before you can insert text into the file you are creating, you must put vim into Input mode. Type the letter **i** (Input) to put vim into Input mode.

   c. With vim in Input mode, type a couple of short lines, ending each line with a RETURN.

   d. Before you can give a command to exit from vim, you must put vim into Command mode. Press ESCAPE to put vim into Command mode.

   e. With vim in Command mode, give the command **ZZ** (type an uppercase **Z** twice) to write the new file to disk and exit from vim. If you did not press ESCAPE before typing **ZZ**, the Zs will appear in your text: Use BACKSPACE to erase them, press ESCAPE, and type **ZZ** again.

3. In its simplest form, the ls utility (Sobell, page 221) lists the names of files in the working directory (Sobell, page 143). After creating a file as explained in step 2, ls will list the name of that file. Use ls to list the names of the files in the working directory. In addition to **practice**, ls lists the directories that Fedora puts there when it sets up a user.

   ```
   $ ls
   ```

   When you call ls with an argument (a word following ls and separated from ls by a SPACE), ls displays the name of the file named by the argument or displays an error message if the file does not exist. Call ls with the name of the file created in step 2 and the string **xxxx** (you must separate each argument from the next by a SPACE). What happens?

```
$ ls practice xxxx
ls: cannot access xxxx: No such file or directory
practice
```

4. As opposed to ls, which lists the *name* of a file, the cat utility (Sobell, page 216) displays the *contents* of a file. Use the cat utility to display the contents of the file you just created.

```
$ cat practice
This is a small text file that I created
using a text editor.
```

5. The cp utility (Sobell, page 224) makes a copy of a file. Use cp to make a copy of the file you just created. Name the copy **practice2**.

```
$ cp practice practice2
```

6. The rm utility (Sobell, page 222) removes (deletes) a file. Use rm to remove the file you created using vim; do not remove the copy of this file you made in the previous step.

```
$ rm practice
```

7. After removing the file you created using vim, what do ls (with no arguments) and cat (with an argument of **practice**) show when you list the name of and contents of that file?

   Because that file no longer exists, ls does not list its name and cat reports that there is no such file; ls does show the copied file.

8. The less utility (Sobell, page 220), which was introduced in an earlier lab, displays a file one screen at a time. Use the less utility to display a long file, such as **/etc/services**, one screen at a time, and then exit from less.

```
$ less /etc/services
```

   Press the SPACE bar to display subsequent screens of text and press **q** to exit from less.

9. By default, the head utility (Sobell, page 235) displays the first ten lines of a file. Use head to display the first ten lines of the **/etc/services** file.

```
$ head /etc/services
```

10. By default, the tail utility (Sobell, page 241) displays the last ten lines of a file. Use tail to display the last ten lines of the **/etc/services** file.

```
$ tail /etc/services
```

11. The hostname utility (Sobell, page 219) displays the name of the system you are working on. Use hostname to display the name of the system you are working on.

```
$ hostname
```

12. The mv utility (Sobell, page 220) renames a file. Use mv to rename the copy of the file you made in step 5. Make the new name of the file **practice**.

```
$ mv practice2 practice
```

13. The lpr utility (Sobell, page 235) sends a file to the printer. Use lpr to print the file you renamed in step 12. With no printer set up lpr displays an error message.

```
$ lpr practice
```

14. The grep utility (Sobell, page 232) searches for a string of characters in a file. Use grep to display all lines that contain a string (such as **small**) in the file you renamed in step 12. Choose a string that is in your **practice** file.

```
$ grep small practice
```

15. Using the vim editor, create a file named **days** that holds the names of the days of the week, in calendar order. Use cat to display the file.

```
$ cat days
Monday
Tuesday
Wednesday
Thursday
Friday
Saturday
Sunday
```

16. The sort utility (Sobell, page 239) displays a file in alphabetical order. Use sort to display the **days** file you created in step 15 in alphabetical order.

```
$ sort days
Friday
Monday
Saturday
Sunday
Thursday
Tuesday
Wednesday
```

17. The file utility (Sobell, page 229) identifies the contents of a file. Use file to determine the type of the **days** file you created in step 15.

```
$ file days
days: ASCII text
```

Which types of files are in the **/etc** directory?

```
$ file /etc/*
/etc/abrt:                    directory
/etc/adjtime:                 ASCII text
/etc/aliases:                 ASCII text
/etc/aliases.db:              regular file, no read permission
/etc/alsa:                    directory
```

```
/etc/alternatives:            directory
/etc/anacrontab:              ASCII text
/etc/asound.conf:             ASCII text
/etc/at.deny:                 very short file (no magic)
/etc/atmsigd.conf:            ASCII text
...
```

DELIVERABLES

This lab gives you practice creating files using the vim editor and an understanding of how to use a few common utilities in their simplest forms.

LAB 6: THE LINUX FILESYSTEM (10 MINUTES)

OBJECTIVES

In this lab you will learn the concepts of home directory, working directory, absolute pathname, and relative pathname. You will learn to use cd to make another directory the working directory, pwd to display the name of the working directory, rm to delete an ordinary file, and mkdir/rmdir to create and remove directory files.

SETUP

- A Fedora 19 installation that includes a nonprivileged user named **student**.

READING

Read the following sections:

- pwd on page 143 of Sobell
- “The Hierarchical Filesystem” on page 176 of Sobell
- “Ordinary Files and Directory Files” on page 177 of Sobell
- “Pathnames” on page 181 of Sobell
- “Working with Directories” on page 183 of Sobell
- “mkdir: Creates a Directory” on page 184 of Sobell
- “cd: Changes to Another Working Directory” on page 185 of Sobell
- “rmdir: Deletes a Directory” on page 186 of Sobell
- “Using Pathnames” on page 187 of Sobell
- “mv, cp: Move or Copy Files” on page 187 of Sobell

Make sure you understand these terms: *directory tree*, *directory file*, *ordinary file*, *filename extension*, *hidden filename*, *working directory*, and *home directory*.

PROCEDURE

1. Your home directory is the directory you are working in when you first log in on the system. Up to this point you have only worked in your home directory. When called by itself, the cd (change directory; Sobell, page 185) command makes your home directory the working directory (the directory you are working in). The pwd (print working directory; Sobell, page 143) command displays the name of the working directory. Give a cd command to make sure you are working in your home directory. Then give a pwd command to confirm you are working in your home directory. What is the name of your home directory?

```
$ cd
$ pwd
```

student or /home/student

2. The mkdir (make directory; Sobell, page 184) utility creates a directory. Use mkdir to create a directory named **two** as a subdirectory of your home directory. Use file to make sure **two** exists and that it is a directory.

   ```
   $ mkdir two
   $ file two
   ```

3. An absolute pathname (Sobell, page 181) starts with a slash and traces a path from the root directory to the file identified by the pathname. As step 1 shows, the pwd command displays the absolute pathname of the working directory. When you call cd with an argument of the name of a directory, it makes that directory the working directory. Use cd to make **two** the working directory and then give a pwd command to confirm the name of the working directory.

   ```
   $ cd two
   $ pwd
   ```

4. Use vim to create a file named **fox** in the **two** directory. Use ls to list the name of **fox**.

   ```
   $ ls fox
   fox
   ```

   or

   ```
   $ ls
   fox
   ```

5. When you give ls an argument that is the name of a directory, ls displays the contents of that directory. Give an ls command with an argument of the absolute pathname of **two** (see step 3) to display the contents of the **two** directory. When you use an absolute pathname, it does not matter which directory is your working directory.

   ```
   $ ls /home/student/two
   fox
   ```

6. Make your home directory the working directory. A relative pathname (Sobell, page 182) is a pathname that does not *start* with a slash, it starts from (is relative to) the working directory. Give an ls command with an argument of **two** (a relative pathname) to display the contents of the **two** directory.

   ```
   $ cd
   $ ls two
   ```

7. The rmdir utility removes an empty directory. Show that rmdir cannot remove the **two** directory while it holds a file.

   ```
   $ rmdir two
   rmdir: failed to remove 'two': Directory not empty
   ```

   Remove **fox** from the **two** directory using a relative pathname and then remove the (empty) **two** directory.

   ```
   $ rm two/fox
   $ rmdir two
   ```

Deliverables

This lab gives you practice using the vim, cd, pwd, rm, mkdir, and rmdir utilities.

Lab 7: Customizing the New System and Using root Privileges (20 minutes)

If you have no connection to the Internet

tip If you have no Internet connection, or a poor connection, you must turn off automatic updates so the system does not keep trying to update itself. You must also set up the Install Media DVD as a repository so you can follow the labs that require you to install software. See Lab 45 (page 151) for instructions on performing these tasks.

Objectives

In this lab you will learn about the special powers of running commands with **root** privileges, also called running commands as a privileged user. The lab covers techniques for gaining **root** privileges and running commands as a nonprivileged user other than yourself. It demonstrates the use of the **who am i** and **whoami** commands to display your real and effective UIDs. You will work with **root** privileges to add user accounts and groups to the system.

This lab adds **ben**, **max**, and the **linux** group

In this lab you will add the users named **ben** and **max** as well as the group named **linux**. The users named **ben** and **max** will belong to the group named **linux**.

Setup

- A Fedora 19 installation that includes a nonprivileged user named **student**. You must know the **root** password.

Reading

Read the following sections:

- "Running Commands with root Privileges" on page 422 of Sobell
- "The Special Powers of a Privileged User" on page 422 of Sobell
- "Gaining root Privileges" on page 423 of Sobell
- "Real UID Versus Effective UID" on page 425 of Sobell
- "Using su to Gain root Privileges" on page 425 of Sobell
- "Managing User Accounts from the Command Line" on page 600 of Sobell

Procedure

You type a command following a shell prompt, usually a dollar sign (**$**). The entire command you type on a line is called a *command line*.

A command line can include options and/or arguments. Options modify the behavior of the command; arguments provide data used by the command. Some commands can be run without options or arguments, some with only options or arguments, and some with both options and arguments (Sobell, “Syntax,” page 144).

You must terminate all commands by pressing the RETURN key (Sobell, “Prompts and RETURNs,” page 27.

PART I: WORKING WITH root PRIVILEGES

1. Open a terminal emulator window as explained on page 15 of this Lab Manual.
2. Execute commands one time and then gain **root** privileges and execute the same commands again to demonstrate the difference between real and effective UIDs (user IDs).

   a. Give the **who am i** and **whoami** commands to display your real and effective UIDs, respectively (Sobell, “Real UID Versus Effective UID,” page 425). Because you have not given any commands to change your UIDs, both commands display **student**.

   ```
   $ who am i
   $ whoami
   ```

   b. Give the **who am i** and **whoami** commands while working with **root** privileges (Sobell, “Executing a Single Command,” page 427). You will need to supply the **root** password. Your real UID will display as **student** (**who am i**) and your effective UID will display as **root** (**whoami**).

   ```
   $ su -c 'who am i'
   $ su -c 'whoami'
   ```

   c. Try using the cat utility (Sobell, page 216) to display the contents of the **/etc/shadow** file (Sobell, page 511), which contains encoded passwords. The cat utility displays a **Permission denied** error message because you are not allowed to read the **shadow** file.

   ```
   $ cat /etc/shadow
   cat: /etc/shadow: Permission denied
   ```

   d. Execute the same command using **root** privileges.

   ```
   $ su -c 'cat /etc/shadow'
   ```

PART II: ADDING USER ACCOUNTS TO THE SYSTEM

3. Add user accounts to the system.

   a. Working as yourself, try to add an account for Max to the system using the useradd utility (Sobell, page 600). The shell displays a **Permission denied** error message because you do not have execute permission for useradd.

```
$ useradd max
-bash: /usr/sbin/useradd: Permission denied
```

b. Working with **root** privileges, add accounts for Max (username **max**) and Ben (username **ben**) to the system.

```
$ su -c 'useradd max'
$ su -c 'useradd ben'
```

c. Working with **root** privileges, use the passwd utility (Sobell, page 137) to add a password of **fit714tree** for Max and Ben. (Normally, different users would have different passwords; these labs assign Sam, Max, and Ben the same password as **student** to keep things simple.) When you give each of the following commands, su first asks you to enter the **root** password and then passwd asks you to enter the new user password two times.

```
$ su -c 'passwd max'
Password: bird482dog
Changing password for user max.
New password: fit714tree
Retype new password: fit714tree
passwd: all authentication tokens updated successfully.

$ su -c 'passwd ben'
...
```

PART III: ADDING GROUPS TO THE SYSTEM

4. Add groups to the system and add users to the groups.

a. Working with **root** privileges, use the groupadd utility (Sobell, page 601) to add the group named **linux** to the **/etc/group** file (Sobell, page 506).

```
$ su -c 'groupadd linux'
```

b. Working with **root** privileges, use the usermod utility (Sobell, page 601) with the **–aG** options to add Ben and Max to the **linux** group.

```
$ su -c 'usermod -aG linux max'
$ su -c 'usermod -aG linux ben'
```

PART IV: WORKING AS A DIFFERENT USER

5. Work as a different user without logging out and logging back in. This technique is useful for security testing and will be used throughout the labs.

Give the command **su – max** to spawn a login shell for Max; enter Max's password (**fit714tree**) in response to the prompt. Next, display your real and effective UIDs as well as the name of the working directory. Give an **exit** command to revert to your original shell and then display your real UID, effective UID, and the name of the working directory again. Your effective UID and the working directory changes, but your real UID remains the same.

```
$ su - max
[max@linux ~]$ who am i
[max@linux ~]$ whoami
[max@linux ~]$ pwd
[max@linux ~]$ exit
$ who am i
$ whoami
$ pwd
```

Deliverables

Two new user accounts (**max** and **ben**) plus one new group named **linux; max** and **ben** belong to the **linux** group.

Lab 8: Updating and Installing Software Using yum

(15 minutes, not counting the time to update the system)

If you do not have Internet access

tip If you do not have Internet access, you must set up the Fedora install DVD or DVD ISO image file as a repository. See Lab 45 (page 151) for instructions before continuing.

Objectives

In this lab you will use yum to update installed software, list installed software packages, and search available software packages.

Setup

- A Fedora 19 installation that includes a nonprivileged user named **student**. You must know the **root** password.
- Access to the Internet *or*
- If you do not have access to the Internet, you must set up the Fedora installation DVD or DVD ISO image file as a repository. See Lab 45 (page 151) for instructions before continuing.

Reading

Read the following sections:

- "Introduction" on page 532 of Sobell
- "JumpStart: Installing and Removing Software Packages Using yum" on page 534 of Sobell
- "Finding the Package That Holds an Application or File You Need" on page 536 of Sobell
- "yum: Keeps the System Up-to-Date" on page 538 of Sobell

Procedure

To update or install software on the local system using yum, a collection of software packages called a *repository* must be available to the local system. During installation, Fedora 19 configures the system to look for a repository at a Fedora mirror site on the Internet. This lab uses the installed default setup.

1. ***This step only: For students with an Internet connection only, update the installed software on the local system. You cannot update installed software from a DVD or DVD ISO image.***

a. Working with **root** privileges, give the following commands to clean yum cache, metadata, and more, and then to update all software packages on the system.

```
$ su -c 'yum clean all'
$ su -c 'yum -y update'
```

The **–y** in the preceding command obviates the need for confirming that you want to install packages while you are running yum. The first update of all software packages on the system takes a long time. The yum utility will update over 1,000 packages. When it is done it will display the message **Complete!** This update includes installation of a new kernel. You must reboot the system to load the new kernel.

b. To reboot the system, click **Sammy Student** at the upper-right corner of the screen and select **Power Off.** From the **Power Off** window click **Restart**. When the system displays the login screen, log in as **Sammy Student** and open a terminal emulator window.

2. Practice installing and removing software (Sobell, page 534) by installing the **httpd** (Apache Web server) and **createrepo** (create repository) packages along with their dependencies and then removing the **python-deltarpm** and **httpd** packages.

```
$ su -c 'yum install httpd'
$ su -c 'yum install createrepo'
$ su -c 'yum remove httpd'
$ su -c 'yum remove python-deltarpm'
```

When you do not include the **–y** (yes) option following **yum**, yum asks if you want to install the package(s); respond by entering **y** and pressing RETURN.

3. Use yum to answer the following questions. You can find sample yum commands in the yum man page and on Sobell, page 539.

a. Which version of bash is installed?

```
$ yum list bash
```

b. Which installed package names begin with the string **kernel**?

```
$ yum list installed 'kernel*'
```

c. Which **system-config** tools are available?

```
$ yum list available 'system-config*'
```

d. Which groups of packages are installed?

```
$ yum group list
```

e. Which packages of the Printing client group are optional?

```
$ yum group info printing
```

f. Which security scanner software is available?

```
$ yum search scanner
$ yum search 'security scanner'
```

g. *Optional:* Which package provides the Apache Web Services? (*Hint:* The Web protocol is HTTP.)

```
$ yum search httpd
```

h. *Optional:* Which package provides the **sshd_config** file? Which package provides the **vsftpd.conf** file?

```
$ yum whatprovides '*/sshd_config'
$ yum whatprovides '*/vsftpd.conf'
```

DELIVERABLES

Practice using yum to install, remove, and query software packages.

LAB 9: EDITING TEXT FILES USING gedit, nano, AND vim (30–40 MINUTES)

OBJECTIVES

In this lab you will learn to use some basic commands of the gedit, nano, and vim editors, with emphasis on vim.

SETUP

- A Fedora 19 installation that includes a nonprivileged user named **student**. You must know the **root** password.

READING

Read the following sections:

- Sobell, "Tutorial: Using nano to Create and Edit a File" (page 270)
- Sobell, "Tutorial: Using vim to Create and Edit a File" (page 262)

PROCEDURE

Each Linux distribution includes several text editors. The default graphical editor with the GNOME desktop is gedit and with KDE is kedit. Both have features similar to the Windows Notepad and WordPad utilities. If the graphical environment is not available, you can use nano, a simple editor that is similar to DOSEdit. You can use vim or emacs to perform more advanced editing including cut and paste, search and replace, and filtering text. This lab explores gedit and nano briefly before focusing on vim.

A graphical text editor has the advantage of being easy to use, similar to text editors in other operating systems, and of allowing you to use the mouse to highlight text to be copied, cut, and pasted. However, many Linux systems, especially servers, do not provide a graphical environment; on these systems you must use a textual editor such as nano or vim.

PART I: EXPLORING gedit

1. Create a new file using the gedit editor. The gedit editor works in a graphical environment (it provides its own window). You do not need to open a terminal emulator window before you can use it.

   a. Open gedit by pressing the SUPER (Windows) key and typing **gedit** as explained in the section on running GUI applications (Lab Manual, page 15). Alternately, you can type **gedit** as a command in a terminal emulator.

   b. Type to add the text **I am creating this file using gedit**.

c. Click **Save** and gedit opens a Save As window that suggests the name **Unsaved Document 1** for the file. Triple click this name to highlight all of it and type **practice.txt** to replace it. Next, select a directory (gedit calls it a *folder*) to save the file in: Under the list labeled **Places** on the left side of the window click the name of your home directory (**student**). Finally, click **Save** at the lower-right corner of the window; gedit saves the file and renames the window with the name of the file: **practice.txt** (~). The tilde (~) indicates that the file is in your home directory.

Note: gedit does not automatically add a filename extension. If you want the file named **practice.txt**, you must specify that entire name, including **.txt**, when saving the file.

d. Exit from gedit and close the gedit window by clicking the **gedit** Application menu (Sobell, Figure 4-7 on page 99) on the top panel and selecting Quit.

e. Open a terminal emulator window (Lab Manual, page 15) and use ls to display the name of the owner of, and file to determine the type of, the file you just created.

```
$ ls -l practice.txt
$ file practice.txt
```

Part II: Exploring nano

2. Modify the *same file* using the nano editor (Sobell, page 270).

a. On the command line in the terminal emulator window you just opened, give the following command

```
$ nano practice.txt
```

The nano takes over the window. It looks as though you are working in a nano GUI window, but you are still working with the CLI.

b. Use the ARROW keys to move the cursor to the end of the file (move the cursor to the line below the one you entered using gedit).

c. Type to add this line to the file: **Next, I edited the file using nano.**

d. Save the file by giving the command CONTROL-O (write out). Press RETURN to keep the same filename.

e. Exit from nano by giving the command CONTROL-X (exit).

Part III: Exploring vim

3. Finally, edit the *same file* using the vim editor (Sobell, page 262).

a. From the command line give the following command

```
$ vim practice.txt
```

As with nano, vim occupies the entire window.

command not found...

tip If you get a **command not found** error, you have not installed the **vim-enhanced** software package; go back to step 1 on page 23 of this Lab Manual and follow the instructions there to install this package before continuing.

b. Use the ARROW keys to place the cursor on the bottom line of text.

c. Press **o** (open; the lowercase "oh" key) to move the cursor to the line below the one it is on and, at the same time, put vim in Input mode. Type to add this line to the file: **Finally, I added this line using vim.** Then press ESCAPE to return vim to Command mode so you can give vim commands instead of entering text.

d. Type **ZZ** (type uppercase **Z** twice) to save the file and exit from vim. If you did not press ESCAPE before typing **ZZ**, the Zs will appear in your text: Use BACKSPACE to erase them, press ESCAPE, and type **ZZ** again.

e. The text in the file will look similar to this:

```
$ cat pactice.txt
I am creating this file using gedit.
Next, I edited the file using nano.
Finally, I added this line using vim.
```

PART IV: EDITING A FILE USING vim

4. Start vim so that you are editing a new file named **pizza**.

```
$ vim pizza
```

5. Type **i** (lowercase **i** for *input;* pages 265 and 268 of Sobell) to put vim in Input mode (Sobell, page 264) and enter the following text, pressing RETURN at the end of each line. Ignore typing mistakes you make for now. Press ESCAPE when you are done typing to put vim back in Command mode (Sobell, page 264).

```
Pizza is an oven-baked, flat, round bread
typically topped with a tomato sauce, cheese
and various toppings. Pizza was originally
invented in Naples, Italy, and the dish has
since become popular in many parts of the world.
(from Wikipedia)
```

6. Use the ARROW keys to move the cursor so that it is over the **o** in **originally.** Press the **x** (delete character; Sobell, page 268) key ten times to delete each of the letters in **originally**. The editor remains in Command mode throughout this step.

7. Search for the word **Italy** by first pressing the **/** (forward slash) key. Pressing this key puts vim in Last Line mode (Sobell, page 265); the cursor moves to the bottom line of the screen. Now type **Italy** and press RETURN to search for

the word **Italy**. Delete the word **Italy** by giving the command **dw** (delete word; Sobell, page 268). The comma and the following SPACE remain. You could remove them using the **x** command, but this step introduces a different command.

Before you can try removing **Italy** another way, you must restore the word you just deleted. Type **u** (undo; Sobell, page 268) to undo the last command you gave; **Italy** reappears. Whereas **dw** deletes a word but not adjacent punctuation, **dW** deletes the word including adjacent punctuation. Give the command **dW**.

8. The **?** (question mark) searches backward for a string of characters the same way the **/** searches forward. Search backward for the word **topped** by typing **?topped** followed by RETURN. The cursor is now on the **t** in **topped**.

9. The **cw** (change word) command removes a word and puts vim in Input mode so you can type a word to replace the original one. You must press ESCAPE to return vim to Command mode when you are finished typing the new word. With the cursor on the first letter of **topped**, give a **cw** command and type the word **covered** followed by an ESCAPE.

10. The **o** (open; Sobell, page 268) command opens a blank line below the line the cursor is on, moves the cursor to the new line, and puts vim in Input mode. The **O** command works the same way except it opens a line above the one the cursor is on. With vim in Input mode and the cursor on a blank line, you can enter as many lines of text as you like. When you are done, press ESCAPE to return vim to Command mode.

    Give an **H** (home) command to move the cursor to the first letter of the document you are editing. Give an **O** (uppercase "oh") command to open a line above the document and type the title **PIZZA** followed by a RETURN and an ESCAPE.

11. Save your work and exit from vim by giving the command **ZZ** (Sobell, page 269). If the **ZZ** appears in the document, vim is in Input mode. Press ESCAPE to put vim back in Command mode, use the **x** command and ARROW keys to remove the **ZZ**, and give another **ZZ** command.

Try other commands from the vim tutorial (Sobell, page 262) as you experiment with the **pizza** file, and create other files to work with.

Part V: Learn to Use vim Online Help

12. This section explains how to use vim online help.

    a. Start vim without specifying a file on the command line (do not give vim an argument); the screen will look like Figure 7-2 on page 264 of Sobell.

    b. Type **:help** and press the RETURN key to display information about vim help. When you type **:**, vim moves the cursor to the last line of the screen (Last Line mode) to give you room to type a command. You must terminate

Last Line mode commands by pressing the RETURN key. With help information displayed, use the ARROW keys to move the cursor down; when you move the cursor past the last line on the screen, the text scrolls to display more information. Type **ZZ** to exit from the help screen and return to the file you were editing. Alternately, you can type **:q**RETURN to close the help window and return to the file you were editing.

c. Type **:help /** (remember to press RETURN) to display information about the **/** (search) command or type **:help exit** to display information about exiting from vim. In many cases vim help displays so much information about a command that it is hard to follow.

d. After you exit from vim help, you still need to exit from vim.

PART VI: USING vim TO EDIT A STARTUP FILE

13. In this section you will use vim to edit the **.bashrc** startup file (Sobell, page 329) in your home directory (**~/.bashrc**); bash executes the commands in this file each time a new shell starts. The change you make will cause bash to display a greeting each time you start a new shell, such as when you open a terminal emulator window.

a. First, make sure you are working in your home directory. Use the cp (copy) utility to make a backup copy of the **.bashrc** file. You can use cp to restore **.bashrc** from this backup file if you make a mistake and **.bashrc** no longer works as it should.

```
$ cd                       # make sure you are working in
                           # your home directory
$ cp .bashrc bashrc.0      # use cp to make a copy of .bashrc
                           # named bashrc.0
```

b. Open vim to edit the **.bashrc** file in your home directory.

```
$ cd          # make sure you are working in your home directory
$ vim .bashrc # use vim to edit .bashrc
```

c. Move the cursor to the last line displayed on the screen. That line will be a comment:

```
# User specific aliases and functions
```

Type **L** (last) or use the ARROW keys.

d. Open a line below the line the cursor is on and put vim in Input mode.

Type **o**.

e. The echo utility copies its arguments to the screen. Type a command that copies the string **Good morning, Sam :**) to the screen, replacing **Sam** with your name. Surround the string (the arguments, not the command) with single quotation marks.

```
echo 'Good morning, Sam :)'
```

f. Return vim to Command mode.

Press ESCAPE.

g. *ONLY* if you made a mistake, press ESCAPE and type **:q!** to exit from vim without making any changes to the **.bashrc** file. Otherwise, save the changes to the file and exit from vim.

Type **ZZ**.

h. Close the terminal emulator window (enter an **exit** command) and open a new terminal emulator window. Does the shell display your message?

PART VII: *OPTIONAL:* USE vimtutor TO LEARN MORE ABOUT vim

Give the command **vimtutor** to start vimtutor. Read at least through lesson 1.2 that explains how to exit from vimtutor.

The **vim-enhanced** package must be installed for vimtutor to work; see step 1 on page 23 of this Lab Manual.

DELIVERABLES

An understanding of how to use the gedit, nano, and vim editors to create and modify files, and a modified **~/.bashrc** startup file.

LAB 10: THE LINUX FILESYSTEM AND COMMON COMMAND-LINE UTILITIES (20 MINUTES)

OBJECTIVES

In this lab you will learn to use Linux command-line utilities to browse the filesystem, determine the type of a file, display text files, and use the online help facilities. At the same time you will learn about the Linux hierarchical file structure.

SETUP

- A Fedora 19 installation that includes a nonprivileged user named **student**.

READING

Read the following sections:

- "Working from the Command Line" on page 119 of Sobell
- "Running Commands from the Command Line" on page 119 of Sobell
- "The Shell" on page 121 of Sobell
- "Running Basic Command-Line Utilities" on page 125 of Sobell
- "man: Displays the System Manual" on page 128 of Sobell
- "info: Displays Information About Utilities" on page 131 of Sobell
- "Special Characters" on page 142 of Sobell
- "Ordinary Files and Directory Files" on page 143 of Sobell

PROCEDURE

1. Open a terminal emulator window (Lab Manual, page 15).
2. The touch utility (Sobell, page 127) takes a filename as an argument. This utility updates the modification and access times of an existing file or creates a new, empty file if the file does not exist. Use touch to create empty files named **myfile** and **memo.1024** in the working directory:

```
$ touch myfile              # create myfile
$ touch memo.1024           # the period is just another character
                            # in the filename
```

3. Give the following commands and observe the behavior with each combination of options and arguments.

a. Use the ls utility (Sobell, page 125) to list the names of the files in the working directory. The ls utility will display the names of the two files you just created plus other files in the working directory.

```
$ ls
```

b. Each file has permissions, an owner, a group, a size, and a modification date and time. Use ls with the **–l** option (long; Sobell, page 191) to display these characteristics for all files in the working directory. The size of the two files you created using touch is zero because touch creates empty files.

```
$ ls -l
```

c. Without an argument, ls lists the files in the working directory. With an argument that is the name of a directory, ls lists the contents of the directory specified by the argument. List the contents of the **/tmp** directory by specifying that directory as an argument to ls.

```
$ ls /tmp
```

d. The ls utility accepts multiple arguments. With a single command, display a list of the files in the **/tmp** and **/var** directories. Display the ownership and size of each file.

```
$ ls -l /tmp /var
```

e. Display information about the **/tmp** and **/var** directories (not the files in the directories).

```
$ ls -ld /tmp /var
```

4. Which utility displays the absolute pathname of the working directory? What is the absolute pathname of the working directory you are working in right now?

```
$ pwd
```

The answer varies depending on your username and the directory you are working in. If you are working in your home directory as the user named **student**, the answer is **/home/student**.

5. Make the **/tmp** directory the working directory and verify the name of the working directory.

```
$ cd /tmp
$ pwd
```

6. Which is the quickest way to make your home directory the working directory?

```
$ cd
```

7. Explore the important standard directories and files listed starting at Sobell, page 189. Make use of cd (Sobell, page 185), ls (Sobell, page 221), pwd (Sobell, page 143), and absolute and relative pathnames (Sobell, page 181).

   a. Use an absolute pathname to make the directory that contains system log files the working directory and then verify the pathname of the working directory.

   ```
   $ cd /var/log
   $ pwd
   ```

   b. Use an absolute pathname to make your home directory the working directory and then verify the pathname of the working directory.

   ```
   $ cd /home/student
   $ pwd
   ```

   c. Starting with your home directory as the working directory, use a relative pathname to make the **/etc** directory the working directory.

   ```
   $ cd ../../etc
   ```

   d. Starting with the **/etc** directory as the working directory, use a relative pathname to list the contents of the directory that contains the OpenSSH configuration files.

   ```
   $ ls ssh
   ```

8. Linux does not rely on filenames or filename extensions to determine the type of a file. Use the file utility (Sobell, page 156) to determine the type of the following files.

   a. **/etc/passwd**

      ASCII text

   b. **/usr/bin/passwd**

      setuid ELF 32-bit LSB shared object (executable)

   c. **/var/log**

      directory

   d. **/usr/share/man/man1/ls.1.gz**

      gzip compressed data

   e. **/dev/tty1**

      character special

   f. **/dev/sda1**

      block special

   g. **/dev/cdrom**

symbolic link

h. **/usr/share/magic**

symbolic link

i. **/usr/share/pixmaps/faces/sky.jpg**

JPEG image data

9. You can display text files using cat (Sobell, page 126), head (page 235), tail (page 241), and less (page 220). Choose the best utility to

a. Display the contents of the **/etc/issue** file.

```
$ cat /etc/issue
```

b. Display the contents of the **/etc/sysconfig/network** file.

```
$ cat /etc/sysconfig/network
```

c. Display the first few lines of the **/etc/passwd** file.

```
$ head /etc/passwd
```

d. Determine the last word in the **/usr/share/dict/linux.words** file.

```
$ tail /usr/share/dict/linux.words
```

e. Display the **/etc/profile** file one page at a time.

```
$ less /etc/profile
```

(*Hint:* Use **q** to quit less.)

10. Display the **/etc/passwd** file using less. Do not exit from less.

```
$ less /etc/passwd
```

a. Display the end of the file by giving the command **G**.

b. Search backward for the string **games** by giving the command **?games**.

c. Display the beginning of the file by giving the command **g**.

d. Search for the string **bash** by giving the command **/bash**.

e. Find the next occurrence of the string **bash** by giving the command **n**.

f. Exit from less by giving the command **q**.

11. Use man and info to find the answer to the following questions.

a. Which ls option sorts the output by modification time?

The **–t** option sorts the output of ls by time. It is easiest to see the effect of this option when you combine it with the **–l** option:

```
$ ls -lt /etc
```

b. Which option to head displays the first five lines of a file instead of the default ten lines? Use this option to display the first five lines of the **/etc/passwd** file.

```
$ head -n 5 /etc/passwd
```

or simply

```
$ head -5 /etc/passwd
```

c. Which option to tail continues to display a file as additional content is added to the file? (*Hint:* Search the tail man page for **follow**.) Use this option to watch the **/var/log/messages** file grow. How do you exit from tail?

```
$ tail -f /var/log/messages
```

Use the interrupt key (CONTROL-C) to exit from tail.

d. Which option to cal displays three months instead of the default current month?

```
$ cal -3
```

e. Use whatis to display a list of man pages that have the string **passwd** in their short descriptions.

```
$ whatis passwd
```

f. Display the man page for the file named **passwd** (not the utility).

```
$ man 5 passwd
```

g. Use **man** with the **–k** option to find a utility that shows who is logged in on the system. A **man –k** command yields the same output as apropos.

```
$ man -k who
```

h. Which command reports on free disk space? (*Tip:* When you want to search for a string of characters that includes a SPACE, put quotation marks around it: **"disk space"**.)

```
$ man -k "disk space"
$ df
```

i. The end of the man page for df has a SEE ALSO referencing info. Explore the df info page. See Sobell, page 131 for instructions on using info.

```
$ info coreutils 'df invocation'
```

12. *Optional:* Learn more about using the info utility by reading the info page on info. (Give the command **info info**.)

Deliverables

Practice using the CLI to issue commands, increased familiarity with a few common commands, and an understanding of the Linux hierarchical filesystem.

Lab 11: More Command-Line Utilities (20 minutes)

Objectives

This lab builds on the previous lab. In this lab you will learn more about working with both regular files and directory files and about compressing files.

Setup

- A Fedora 19 installation that includes a nonprivileged user named **student**.

Reading

Read the following sections:

- Sobell, "rm: Removes a File (Deletes a Link)" (page 222)
- Sobell, "cp: Copies Files" (page 224)
- Sobell, "mv: Renames or Moves a File" (page 237)
- Sobell, "Compressing and Archiving Files" (page 245)
- Sobell, "locate: Searches for a File" (page 256)

Procedure

1. Open a terminal emulator window (Lab Manual, page 15).
2. Working as yourself, create a directory named **Unit2** as a subdirectory of your home directory. Before you create the directory, use cd to make your home directory the working directory and then use pwd to display the name of the working directory.

```
$ cd
$ pwd
$ mkdir Unit2
```

3. Create subdirectories under **Unit2** named **memos** and **reports**.

```
$ mkdir Unit2/memos
$ mkdir Unit2/reports
```

or

```
$ cd Unit2
$ mkdir memos reports
```

4. Make sure your home directory is the working directory and then use touch (Sobell, page 127) to create these files: **memo.one, memo.two, memo.three, report.jan**, and **report.feb**, and **report.mar**. Create the three report files using a single command (touch accepts multiple arguments).

```
$ cd
$ touch memo.one
$ touch memo.two
$ touch memo.three
$ touch report.jan report.feb report.mar
```

5. Copy the **memo.one** file to the **Unit2/memos** directory.

```
$ cp memo.one Unit2/memos
```

6. Using a single command, copy the **memo.two** file to the **Unit2/memos** directory, making the filename of the copied file **memo.2**.

```
$ cp memo.two Unit2/memos/memo.2
```

7. *Move* the **memo.three** file to the **Unit2/memos** directory.

```
$ mv memo.three Unit2/memos
```

8. *Move* the three report files to the **Unit2/reports** directory.

```
$ mv report.jan report.feb report.mar Unit2/reports
```

9. Make the **Unit2/reports** directory the working directory. Before modifying the **report.jan** file, make a backup copy of the file with a **.orig** filename extension.

```
$ cd Unit2/reports
$ cp report.jan report.jan.orig
```

10. Copy the **/etc/passwd** file to **Unit2/reports/report.jan**, overwriting the **report.jan** file. Use a cat command before and after you copy the file to verify that you overwrote the file.

```
$ cat report.jan
$ cp /etc/passwd report.jan
$ cat report.jan
```

11. Copy the **/etc/hosts** file to **Unit2/reports/report.feb**. Have the utility you use to copy the file ask for confirmation before overwriting **report.feb**.

```
$ cp -i /etc/hosts report.feb
```

12. Remove the remaining memo files from your home directory.

```
$ cd
$ rm memo.one memo.two
```

13. Remove the **report.jan** file. Have the utility you use to remove the file ask for confirmation before removing **report.jan.**

```
$ rm -i Unit2/reports/report.jan
```

14. Remove the **memos** directory including all files in the directory.

```
$ rm -r Unit2/memos
```

or

```
$ rm Unit2/memos/memo.one
$ rm Unit2/memos/memo.2
$ rm Unit2/memos/memo.three
$ rmdir Unit2/memos
```

15. Use locate to find a file that contains the string **chess** in its name.

```
$ locate chess
```

16. Use locate to find a file that contains the string **sky** in its name.

```
$ locate sky
```

17. Copy the **/usr/share/dict/linux.words** file to your home directory.

```
$ cd
$ cp /usr/share/dict/linux.words .
```

or

```
$ cp /usr/share/dict/linux.words /home/student
```

or

```
$ cp /usr/share/dict/linux.words ~
```

a. What is the size of the **linux.words** file in bytes?

```
$ ls -l linux.words
```

The file is approximately 5 megabytes (5,000,000 bytes) long.

b. Compress the **linux.words** file using gzip. What is the size of the compressed file in bytes?

```
$ gzip linux.words
$ ls -l linux.words.gz
```

The file is approximately 1.5 megabytes (1,500,000 bytes) long.

c. Uncompress the **linux.words** file. What is the size of the uncompressed file in bytes? Is it the same size as the original file?

```
$ gunzip linux.words.gz
$ ls -l linux.words
```

The file is approximately 5 megabytes (5,000,000 bytes) long. Yes, it is the same size as the original file.

d. Compress the **linux.words** file using **bzip2**. What is the size of the compressed file in bytes?

```
$ bzip2 linux.words
$ ls -l linux.words.bz2
```

The file is approximately 1.7 megabytes (1,700,000 bytes) long.

e. Uncompress the **linux.words** file. What is the size of the uncompressed file in bytes?

```
$ bunzip2 linux.words.bz2
$ ls -l linux.words
```

The file occupies approximately 5 megabytes (5,000,000 bytes).

18. *Optional:* Repeat the preceding compression exercises using other types of files, such as a JPG file from the **/usr/share/pixmaps/faces** directory and an executable file from the **/usr/bin** directory. Copy the files to your home directory before attempting to compress them.

Deliverables

Files organized in the **Unit2** subdirectory; practice working with regular and directory files and compressing files.

Lab 12: File Access Permissions (20 minutes)

Objectives

In this lab you will use ls to display, and chmod to change, file and directory access permissions. You will also use the umask utility to display and change the file permissions mask.

Setup

- A Fedora 19 installation that includes nonprivileged users named **student**, **max**, and **ben**. You must know the **root** password.

Reading

Read the following sections:

- "Access Permissions" on page 191 of Sobell
- "ls –l: Displays Permissions" on page 191 of Sobell
- "chmod: Changes File Access Permissions" on page 193 of Sobell
- "umask: Specifies the File Permission Mask" on page 469 of Sobell

Procedure

1. Use touch to create a file named **dog** in the working directory. Use ls to confirm the file was created.

```
$ touch dog
$ ls dog
```

2. Use **ls –l** (Sobell, page 191) to display the permissions of the file you just created. Who owns the file? Which group is it associated with? Which permissions does the owner of the file have? The group? Others?

```
$ ls -l dog
-rw-rw-r--. 1 student student 0 Dec 17 19:13 dog
```

Based on the output of ls, the owner of the file and the group the file is associated with have read and write permissions while others have only read permission. The user named **student** owns the file and the file is associated with the **student** group. Your permissions may be different.

3. Display the permissions of **/bin/bash**. Who owns the file? Which group is it associated with? Which permissions does the owner of the file have? The group? Others? Which permissions apply to you?

```
$ ls -l /bin/bash
-rwxr-xr-x. 1 root root 917576 Mar 11  2013 /bin/bash
```

Based on the output of ls, the owner of the file has read, write, and execute permissions while the group and others have only read and execute permissions. The owner of the file is **root** and the file is associated with the **root** group. The permissions for others (read and execute) apply to you (working as **student**) because you are not the user named **root**, you are not a member of the group named **root**, and you are not working with **root** privileges.

4. Only the owner of a file (or a user working with **root** privileges) can change the permissions of a file. Using numeric arguments to chmod (Sobell, page 193), change the permissions of the file you created in step 1 so the owner has read and write permissions and the group and others have no permissions. Next change the permissions so the owner, the group, and others have only read permissions. Display the permissions of the file before you start and after each change.

```
$ ls -l dog
-rw-rw-r--. 1 student student 0 Dec 17 19:13 dog
$ chmod 600 dog
$ ls -l dog
-rw-------. 1 student student 0 Dec 17 19:13 dog
$ chmod 444 dog
$ ls -l dog
-r--r--r--. 1 student student 0 Dec 17 19:13 dog
```

5. Determine the permissions of various files and directories on the system. Fill in the following chart:

Filename	Readable by	Writable by	Executable/ searchable by	Symbolic type and permissions	Octal permissions
/etc/passwd	owner (**root**), group (**root**), and others	owner (**root**)	none	-rw-r--r--	644
/etc/shadow	none	none	none	----------	000
/etc/cups/cupsd.conf	owner (**root**) and group (**lp**)	owner (**root**)	none	-rw-r-----	640
/var/log/audit	owner (**root**)	owner (**root**)	none	-rw-------	600
/var/log/cups	owner (**lp**), group (**sys**), and others	owner (**lp**)	owner (**lp**), group (**sys**), and others	drwxr-xr-x	755
/etc/cups/ssl	owner (**root**)	owner (**root**)	owner (**root**)	drwx------	700
/var/spool/mail	owner (**root**), group (**root**), and others	owner (**root**) and group (**root**)	owner (**root**), group (**root**), and others	drwxrwxr-x	775

Filename	Readable by	Writable by	Executable/ searchable by	Symbolic type and permissions	Octal permissions
/var/spool/cron	owner (**root**)	owner (**root**)	owner (**root**)	drwx------	700
/dev/tty1	owner (**root**)	owner (**root**) and group (**tty**)	none	crw--w----	620
/bin/chmod	owner (**root**), group (**root**), and others	owner (**root**)	owner (**root**), group (**root**), and others	-rwxr-xr-x	755
/usr/bin/procmail	owner (**root**), group (**mail**), and others	owner (**root**)	owner (**root**), group (**mail**), and others	-rwxr-xr-x	755
/usr/bin/locate	owner (**root**)	owner (**root**)	owner (**root**), group (**root**), and others (with **root** privileges)	-rwx--s--x	2711
/usr/bin/crontab	owner (**root**), group (**root**), and others	owner (**root**)	owner (**root**), group (**root**), and others (with **root** privileges)	-rwsr-sr-x	6755

6. Change permissions of files.

   a. Make the **Unit2/reports/report.feb** file readable by the owner and group, writable by the owner only, and not accessible by others.

```
$ chmod 640 Unit2/reports/report.feb
```

   or

```
$ chmod o-r,g-w Unit2/reports/report.feb
```

   b. Create a subdirectory in **/tmp** named **shared**. The file should be owned by **ben** and associated with the group named **linux**. The user named **ben** and the owner of the directory and members of the **linux** group should have read, write, and execute (search) permission to the directory. Other users should have no access. (*Tip:* Run these commands as the user **ben.**) Working as **student** once again, list the permissions and ownership information of the new directory.

```
$ su - ben
[ben@linux ~]$ mkdir /tmp/shared
[ben@linux ~]$ chgrp linux /tmp/shared
[ben@linux ~]$ chmod 770 /tmp/shared
[ben@linux ~]$ exit
$ ls -ld /tmp/shared
drwxrwx---. 2 ben linux 40 Mar  6 10:47 /tmp/shared
```

c. Working as the user named **max**, create a file named **max1** in the **/tmp/shared** directory. Then, once again working as the user named **student**, display your effective UID and then attempt to create a file named **student1** in **/tmp/shared**. The touch utility will display a **Permission denied** error.

```
$ su - max
[max@linux ~]$ touch /tmp/shared/max1
[max@linux ~]$ exit
$ whoami
student
$ touch /tmp/shared/student1
touch: cannot touch '/tmp/shared/student1': Permission denied
```

d. Make sure you are working as **student**. If you are not, give an **exit** command and check again.

```
$ whoami
```

7. The permissions of a new file are determined by the file permissions mask that is set by the umask utility (Sobell, page 469).

a. Working as the user named **student**, display the value of the file permissions mask.

```
$ umask
```

b. Create a new file named **mask.one** and display the permissions.

```
$ touch mask.one
$ ls -l mask.one
```

c. Change the file permissions mask to 077 and then create a file named **mask.two**. Display the permissions of the new file.

```
$ umask 077
$ touch mask.two
$ ls -l mask.two
```

d. Create a new directory named **masks** and display the permissions of the new directory.

```
$ mkdir masks
$ ls -ld masks
```

8. *Optional:* Repeat the procedure in step 7 for other file permissions masks such as 027, 022, and 777.

9. Reset the permissions mask to 002.

```
$ umask 002
```

Deliverables

This lab gives you practice using **ls –l**, touch, chmod, and umask to gain an understanding of file ownership and permissions.

Lab 13: System Administration Utilities (30–40 minutes)

Objectives

In this lab you will explore commands used to monitor and manage a Linux system. Use the man and info utilities or the Sobell text to look up commands, arguments, and options rather than memorizing them.

Setup

- A Fedora 19 installation that includes a nonprivileged user named **student**. You must know the **root** password.

This lab adds **casey** and the **staff** group

In this lab you will add the user named **casey** and the group named **staff.** The users named **ben** and **casey** will belong to the group named **staff.** See "Nonprivileged users" on page 5 of this Lab Manual for a list of labs wherein these students and associated groups were created.

Reading

Read the following sections:

- "grep: Searches for a Pattern in Files" on page 232 of Sobell
- "du: Displays Disk Usage Information" on page 523 of Sobell
- "free: Displays Memory Usage Information" on page 253 of Sobell
- "Runlevels" on page 440 of Sobell
- "Runlevels" on page 449 of Sobell
- "df: shows where directory hierarchies are mounted" on page 804 of Sobell

Procedure

1. Use the grep utility (Sobell, page 232) to accomplish the following tasks:

   a. Display each line in the **/etc/passwd** file that contains the string **student**.

   ```
   $ grep student /etc/passwd
   ```

   b. Verify that **ben** is a member of more than one group by displaying all occurrences of the string **ben** in the **/etc/group** file.

   ```
   $ grep ben /etc/group
   ```

   c. Search all files and directories whose pathnames *start* with **/etc/yum** for the string **yum**. Include all subdirectories in the search.

   ```
   $ grep -R yum /etc/yum*
   ```

d. List the filename of each file whose pathname *starts* with **/etc/yum** that contains at least one occurrence of the string **yum**.

```
$ grep -R -l yum /etc/yum*
```

e. What are the line numbers of the lines in the **/etc/group** file that reference **max**?

```
$ grep -n max /etc/group
```

f. List all accounts in the **/etc/passwd** file that do *not* use the bash shell.

```
$ grep -v bash /etc/passwd
```

g. How many accounts in the **/etc/passwd** file do *not* use the bash shell?

```
$ grep -vc bash /etc/passwd
```

h. Display all references to the **PATH** variable (Sobell, page 359) in **/etc/profile**.

```
$ grep PATH /etc/profile
```

i. The **/etc/profile** script modifies the **PATH** variable using a **pathmunge** command. List all references to both **PATH** and **pathmunge** with one query. (*Hint:* Search for uppercase and lowercase **path**.)

```
$ grep -i path /etc/profile
```

2. Use the man pages and the text for useradd (Sobell, page 600), usermod (Sobell, page 601), groupadd (Sobell, page 601), and passwd (Sobell, page 137) to form commands to complete the following tasks. (*Tip:* You will need to work with **root** privileges.)

a. Create a group named **staff** with a GID (group ID) of 1004.

```
$ su -c 'groupadd -g 1004 staff'
```

b. Create a user named **casey** with a supplementary group of **staff**.

```
$ su -c 'useradd -G staff casey'
```

c. Add **ben** to the **staff** group.

```
$ su -c 'usermod -aG staff ben'
```

d. Set the password for **casey** to **fit714tree**.

```
$ su -c 'passwd casey'
```

3. Use du (Sobell, page 523) to display filesystem information and use df (Sobell, page 804) to check disk space usage from the command line.

a. How much disk space does Ben's home directory occupy?

```
$ su -c 'du -sh /home/ben'
```

About 32 kilobytes.

b. How much space does the **/usr/bin** directory occupy?

```
$ du -sh /usr/bin
```

About 137 megabytes.

c. How much disk space is available in the **root** (**/**) filesystem? Specify the answer in megabytes (MB) or gigabytes (GB).

```
$ df -h /
```

About 9.4 gigabytes.

d. Give a command that displays the percentage of space being used in, and the filesystem type of, the filesystem mounted at **/home**.

```
$ df -T /home
```

or

```
df -Th /home
```

4. List the users logged in on the system.

```
$ users
```

or

```
$ who
```

or

```
$ w
```

5. Display information about the system.

a. Display the amount of memory in the system (Sobell, page 253).

```
$ free -h
```

b. Display the amount of time the system has been running (since the last boot), the number of users who are logged in, and what each user is doing (Sobell, page 254).

```
$ w
```

c. Display the username you are logged in as (your real UID; Sobell, page 254).

```
$ who -m
```

or

```
$ who am i
```

d. Display the current target unit (runlevel; pages 440 and 449 of Sobell).

```
$ runlevel
```

or

```
$ who -r
```

Deliverables

Practice using system administration command-line utilities.

LAB 14: WORKING WITH ACLS (15–20 MINUTES)

OBJECTIVES

In this lab you will extend traditional Linux file permissions by using ACLs (Access Control Lists).

SETUP

- Fedora 19 installation with several nonprivileged users (**student**, **ben**, **max**, and **casey**). See "Nonprivileged users" on page 5 of this Lab Manual for a list of labs wherein these students and associated groups were created.

This lab adds **rose**

In this lab you will add the user named **rose**.

READING

Read the following section:

- "ACLs: Access Control Lists" on page 198 of Sobell

PROCEDURE

1. Make sure the file permissions mask is set to 002.

   ```
   $ umask
   0002
   ```

   If the file permissions mask is not set to 002, reset it.

   ```
   $ umask 002
   ```

2. Working as the user named **student**, create a directory in **/tmp** named **campaign**.

   ```
   $ mkdir /tmp/campaign
   ```

3. Use vim to create a short file named **vis** in the **campaign** directory. It does not matter what text you put in the file. Use chmod (Sobell, page 193) to give the owner, group, and other users read and write access to **vis**.

   ```
   $ vim /tmp/campaign/vis
   ```

   ```
   $ chmod 666 /tmp/campaign/vis
   ```

   or

   ```
   $ chmod a=rw /tmp/campaign/vis
   ```

4. Display the standard access permissions for the **campaign** directory. What access do these permissions allow?

```
$ ls -ld /tmp/campaign
drwxrwxr-x. 2 student student 60 Dec  6 18:31 /tmp/campaign
```

The permissions allow **student** and members of the group named **student** to read from, write to, and search the directory, and allow others to read from and search the directory (but not write to it).

5. Display the ACL permissions for the **campaign** directory using getfacl (Sobell, "Working with Access Rules," page 199).

   ```
   $ getfacl /tmp/campaign
   ```

6. Adjust the permissions so that

   a. Members of the group **staff** can read from and search, but not write to, the **campaign** directory. *Hint:* Use chgrp (Sobell, page 195) and chmod (Sobell, page 193).

   ```
   $ su -c 'chgrp staff /tmp/campaign'
   $ chmod 750 /tmp/campaign
   ```

   b. The user **casey** can read from, write to, and search the **campaign** directory. *Hint:* Use setfacl (Sobell, page 200).

   ```
   $ setfacl -m u:casey:rwx /tmp/campaign
   ```

   c. The user **max** can only read from and search the **campaign** directory (use setfacl again).

   ```
   $ setfacl -m u:max:rx /tmp/campaign
   ```

   d. Other users have no access to the directory (use chmod).

   ```
   $ chmod o-rwx /tmp/campaign
   ```

7. Create a new account for Rose to use while testing these permissions (Lab Manual, page 31). Assign Rose a password of **fit714tree**, the same as the other nonprivileged users.

   ```
   $ su -c 'useradd rose'
   $ su -c 'passwd rose'
   ```

8. Test the permissions by working as **student**, **casey**, **max**, and **rose**. For each user, attempt to list the contents of the directory and create a file in the directory.

Exiting from vim when you cannot write a file

tip In an emergency, such as when you cannot exit from vim because you cannot write to a file, you can exit from vim by giving the command **:q!**RETURN. Exiting in this manner does not save the text you just entered into nor the changes you just made to the file.

The following commands show how to work as the user named **casey** and, while doing so, test the permissions for **casey**; repeat this set of commands for each user.

```
$ su - casey
[casey@linux ~]$ ls -l /tmp/campaign
[casey@linux ~]$ vim /tmp/campaign/casey1
[casey@linux ~]$ exit
```

Results will vary by user: Rose should have no access and Max should not be able to write the file.

9. Look at the permissions of the new files that **student** and **casey** created. Can **max** read the files? Can **max** create files in the directory?

```
$ getfacl /tmp/campaign/*
```

Assuming the default umask of 002, **max** can read the files created by other users but not write to those files; **max** cannot create files in the directory.

10. Make sure you are working as **student**. Modify the *directory* ACLs so that **casey** can read and write new files in the directory and **max** can read but not write new files in the directory.

```
$ setfacl -m d:u:casey:rw /tmp/campaign
$ setfacl -m d:u:max:r /tmp/campaign
```

11. Test the new permissions by creating a file in the directory and viewing the ACL rules.

```
$ touch /tmp/campaign/newfile
$ getfacl /tmp/campaign/newfile
```

DELIVERABLES

A shared directory with access controlled by a combination of traditional and ACL rules.

LAB 15: EXPLORING bash SHELL FEATURES (15–20 MINUTES)

OBJECTIVES

In this lab you will explore bash syntax for brace expansion, filename generation, redirection, and pipelines.

SETUP

- A Fedora 19 installation that includes a nonprivileged user named **student**. You must know the **root** password.

READING

Read the following sections:

- "Brace Expansion" on page 405 of Sobell
- "Filename Generation/Pathname Expansion" on page 165 of Sobell
- "Redirection" on page 153 of Sobell
- "Redirecting Standard Error" on page 333 of Sobell
- "Pipelines" on page 158 of Sobell

PROCEDURE

PART I: BRACE EXPANSION

1. Use brace expansion (Sobell, page 405) to create files to work with in this lab. Type in the following command to create 36 files in your home directory.

```
$ touch {report,memo,reminder}_{1,2,3,4}.{new,old,keep}
```

PART II: FILENAME GENERATION

2. Give one mkdir command (Sobell, page 184) to create a directory named **Unit3** with subdirectories named **reports**, **memos**, **backups**, and **keepsakes**.

```
$ mkdir -p Unit3/reports Unit3/memos Unit3/backups Unit3/keepsakes
```

or

```
$ mkdir -p Unit3/{reports,memos,backups,keepsakes}
```

3. Organize the new files as follows.

   a. Copy all the files that end in **keep** to the **keepsakes** directory.

```
$ cp *keep Unit3/keepsakes
```

b. Move all the report files to the **reports** directory.

```
$ mv report* Unit3/reports
```

c. Move all the memo files to the **memos** directory.

```
$ mv memo* Unit3/memos
```

d. Use filename expansion to remove all versions of reminders 1 and 2.

```
$ rm reminder_[12]*
```

e. Use filename expansion to copy the fourth version of the old files to the backups directory.

```
$ cp Unit3/*/*4.old Unit3/backups
```

PART III: REDIRECTION

4. Display all files in the **Unit3** directory tree.

```
$ ls -R Unit3
```

5. Use redirection to create a file named **ls.out** that contains a list of all the files in the **Unit3** directory tree.

```
$ ls -R Unit3 > ls.out
```

6. Use redirection to append the date to the end of the **ls.out** file.

```
$ date >> ls.out
```

7. Redirecting error messages.

a. Use the grep **–R** option to list each occurrence of **student** in all the files that you have read access to in the **/etc** directory and its subdirectories.

```
$ grep -R student /etc
```

b. Modify the preceding grep command so that it redirects **Permission denied** errors to the null device (Sobell, page 158). (*Tip:* Most utilities send error messages to standard error; Sobell, page 333.)

```
$ grep -R student /etc 2> /dev/null
```

c. Instead of listing each line in each file that contains **student**, list only the filenames of files (that you have read access to) in the **/etc** directory hierarchy that contains at least one line with **student**.

```
$ grep -R -l student /etc 2> /dev/null
```

PART IV: PIPELINES

8. The **locate chess** command from step 15 on page 50 of this Lab Manual might have scrolled off the screen. Run the command again and display the output one screen at a time. (*Tip:* Use a pipeline and less; Sobell, page 158.)

```
$ locate chess | less
```

9. Use locate (Sobell, page 256) to list all files with **chess** in their names that are *not* help files (*Hint:* Use a pipeline and grep.)

```
$ locate chess | grep -v help
```

10. *Optional:* Eliminate all files that include the string **jar** in their filenames from the list that step 7c outputs. (*Hint:* Use a pipeline and a second grep command at the end of the command you wrote previously.)

```
$ grep -R -l student /etc 2>/dev/null | grep -v jar
```

DELIVERABLES

Experience using mkdir, cp, mv, and grep, a better understanding of the hierarchical file structure, and organized files with the directory hierarchy recorded in the file named **ls.out**.

Lab 16: Shell Parameters and Variables (15 minutes)

Objectives

In this lab you will learn about user-created variables and keyword variables.

Setup

- A Fedora 19 installation that includes a nonprivileged user named **student**. You must know the **root** password.

Reading

Read the following sections:

- "Parameters and Variables" on page 352 of Sobell
- "User-Created Variables" on page 353
- "Variable Attributes" on page 356
- "Keyword Variables" on page 358 of Sobell
- "PS1: User Prompt (Primary)" on page 361 of Sobell
- "Aliases" on page 392 of Sobell

Procedure

Although variables are mostly used in scripts and read by programs, you can experiment with them on the command line.

1. Assign your name to the variable named **myname** and use echo to display the value of **myname** when it is unquoted, quoted using double quotation marks, and quoted using single quotation marks. ("Parameter substitution" on page 353 of Sobell and "Quoting the **$**" on page 354 of Sobell.)

```
$ myname=Max
$ echo $myname
Max
$ echo "$myname"
Max
$ echo '$myname'
$myname
```

2. Use the readonly (Sobell, page 356) builtin to make the **myname** variable you created in the previous step a readonly variable and then attempt to assign a new value to it. What happens?

```
$ readonly myname
$ myname=Sam
bash: myname: readonly variable
```

3. What is the value of your **HOME** (Sobell, page 358) keyword variable?

```
$ echo $HOME
/home/student
```

Demonstrate that the tilde (~; Sobell, page 359) holds the same value as **HOME**. List the contents of your home directory using a tilde.

```
$ echo ~
/home/student

$ ls ~
```

4. The **PATH** (Sobell, page 359) keyword variable specifies the directories in the order bash should search them when you run a script or program from the command line. What is the value of your **PATH** variable?

```
$ echo $PATH
/usr/local/bin:/usr/bin:/usr/local/sbin:/usr/sbin:/home/student/.local/bin:/home/student/bin
```

Append the absolute pathname of the **myprogs** directory that is a subdirectory of your home directory to the **PATH** variable. What does this change allow you to do more easily? (*Hint:* You can use a variable while defining a variable.)

```
$ PATH=$PATH:/home/student/myprogs
$ echo $PATH
/usr/local/bin:/usr/bin:/usr/local/sbin:/usr/sbin:/home/student/.local/bin:/home/student/bin:/home/stu
dent/myprogs
```

Adding the **~/myprogs** directory to **PATH** allows you to run scripts in that directory by just typing a simple filename (you do not need to type a pathname that includes a slash). However, Fedora sets up **~/bin** for this purpose so you do not need to add **~/myprogs** to **PATH**, you can just use **~/bin**.

5. The **PS1** (Sobell, page 361) keyword variable holds the value of your primary shell prompt. Change the value of this variable so that your prompt is a **$** followed by a SPACE when you are working as yourself and a **#** followed by a SPACE when you are working with **root** privileges.

```
$ PS1='\$ '
```

6. Display the values of the **HOSTNAME**, **MAIL**, **PWD**, and **USER** variables.

```
$ echo $HOSTNAME
$ echo $MAIL
$ echo $PWD
$ echo $USER
```

7. When you give the command **su –**, you spawn a login **root** shell; it is as though you logged in as **root**. When you omit the hyphen (**–**), you still have **root** privileges, but the environment variables are not changed.

Using the commands from step 6 of this lab, fill in the following chart.

Variable	Working as the user named **student**	After gaining **root** privileges using **su** (no dash!)	After gaining **root** privileges using **su –**
HOSTNAME	linux.example.com	linux.example.com	linux.example.com
PATH	/usr/local/bin:/usr/bin:/usr/local/sbin:/usr/sbin:/home/student/.local/bin:/home/student/bin	/usr/local/bin:/usr/bin:/usr/local/sbin:/usr/sbin:/home/student/.local/bin:/home/student/bin	/usr/local/sbin:/usr/local/bin:/sbin:/bin:/usr/sbin:/usr/bin:/root/bin
MAIL	/var/spool/mail/student	/var/spool/mail/student	/var/spool/mail/root
PWD	/home/student	/home/student	/root
USER	student	student	root

8. Modify your shell prompt to include the date, your name, the hostname, the name of the working directory, and the history number. (*Hint:* Give the command **man bash** and search for **PROMPTING**.)

```
$ PS1='[\d][\u@\h \W]\! \$ '
```

9. Create an alias such that when you type **lt**, the command displays a long, recursive listing of all files in the directory (e.g., **lt Unit3** would display a long, recursive listing of the **Unit3** directory hierarchy).

```
$ alias lt='ls -alR'
```

DELIVERABLES

This lab gives you practice working with user-created variables and the **HOME**, **HOSTNAME**, **PATH**, **MAIL**, **PWD**, **USER**, and **PS1** keyword variables as well as practice using the echo utility. It also sets up a customized bash environment.

LAB 17: WRITING AND EXECUTING A SHELL SCRIPT (20 MINUTES)

OBJECTIVES

In this lab you will learn to write and execute a shell script that includes comments. You will use chmod to make the file that holds the script executable and include a line that starts with **#!** in the script to make sure bash executes it. This lab also provides an introduction to positional parameters.

READING

Read the following sections:

- "Writing and Executing a Shell Script" on page 336 of Sobell
- "$1–$***n***: Positional Parameters" on page 1023 of Sobell

PROCEDURE

1. Use vim (Lab Manual, page 37) to create a file named **short** with the following line in it:

   ```
   echo 'hi there'
   ```

2. Use cat to verify the contents of **short** and then try to execute it. Use **ls –l** to display the permissions for **short**. Read the tip on page 338 of Sobell.

   ```
   $ cat short
   echo 'hi there'

   $ ./short
   bash: ./short: Permission denied

   $ ls -l short
   -rw-r--r--. 1 student student 16 10-19 13:31 short
   ```

3. Use chmod (pages 193 and 337 of Sobell) to make the file executable, display the permissions for **short**, and try executing the file again.

   ```
   $ chmod u+x short
   $ ls -l short
   -rwxr--r--. 1 max pubs 16 10-19 13:31 short
   ```

   or

   ```
   $ chmod 755 short
   $ ls -l short
   -rwxr-xr-x. 1 max pubs 16 10-19 13:31 short

   $ ./short
   hi there
   ```

4. Add a line that starts with **#!** (Sobell, page 338) to the beginning of **short** to make sure it is executed by bash.

   ```
   $ cat short
   #!/bin/bash
   echo 'hi there'
   ```

5. Add a comment line (Sobell, page 340) to **short** that explains what the script does.

   ```
   $ cat short
   #!/bin/bash
   # This script sends the string 'hi there' to standard output
   echo 'hi there'
   ```

6. Within a shell script, the shell expands **$1** (a *positional parameter;* Sobell, page 1023) to the first argument on the command line the script was called with. Write and execute a script named **first** that displays (sends to standard output) the first argument on the command line it was called with. Include the **#!** line and a comment. Remember to make the file executable.

   ```
   $ cat first
   #!/bin/bash
   #
   # This script sends its first argument to standard output
   #
   echo $1

   $ ./first Hello
   Hello
   ```

7. The date (Sobell, page 218) utility displays the date and time. Write and execute a shell script named **mine** that displays the date and time, the name of your home directory, and the value of your **PATH** variable.

   ```
   $ cat mine
   #!/bin/bash
   # This script displays time and date,
   # the name of your home directory, and
   # the value of your PATH variable.
   date
   echo $HOME
   echo $PATH

   $ chmod 755 mine

   $ ./mine
   Thu Dec 19 13:07:29 PST 2013
   /home/student
   /usr/local/bin:/usr/bin:/usr/local/sbin:/usr/sbin:/home/student/.loc
   al/bin:/home/student/bin
   ```

8. Write a shell script that copies the file named by its first argument to a file with the same name with the filename extension of **.bak**. Thus, if you call the script with the argument **first** (and a file named **first** exists in the working directory), after the script runs you would have two files: **first** and **first.bak**. Demonstrate that the script works properly.

```
$ cat cptobak
#!/bin/bash
#
# This script copies the file named by its first argument
# to the same name with a filename extension of .bak.
#
cp $1 $1.bak

$ ls fir*
first

$ ./cptobak first

$ ls fir*
first  first.bak
```

9. Read the caution titled "Always quote positional parameters" on page 1023 of Sobell. Use touch to create a file whose name has a SPACE in it. What happens when you give that filename as an argument to **cptobak** from the previous step?

```
$ touch aa\ bb
$ ls -l aa*
-rw-rw-r-- 1 student student 0 Jan 15 16:35 aa bb
$ ./cptobak aa*
cp: target 'bb.bak' is not a directory
```

The shell passes one argument (that includes a SPACE) to **cptobak**: **aa bb**. But when the shell that is running **cptobak** parses **$1**, it expands it to two arguments because it contains an unquoted SPACE; the shell passes to cp four arguments:

```
cp aa bb aa bb.bak
```

When you pass more than two arguments to cp, the last argument must be a directory: cp copies the files named by the first arguments into the directory named by the last argument. If the last argument is not a directory, cp displays an error message.

10. Modify the **cptobak** script from the previous step by quoting the positional parameters in the cp line. Now what happens when you use the script to make a copy of a file with a SPACE in its name?

```
$ cat cptobak
...
cp "$1" "$1.bak"
```

```
$ ./cptobak aa*
$ ls -l aa*
aa bb  aa bb.bak
```

Deliverables

This lab gives you practice writing and executing shell scripts.

Lab 18: Using ssh to Connect to a Remote System (10–15 minutes)

Objectives

In this lab you will set up the **/etc/hosts** file on the local system to make it easier to connect to the classroom computer. Then you will use the ssh utility to connect to the classroom computer and, from the classroom computer, back to the local (student) system.

Setup

- A Fedora 19 installation that includes a nonprivileged user named **student**. By default, Fedora is set up as an OpenSSH server: The local system can accept an ssh connection from a remote system.
- Access to the classroom server. See the following tip.

Reading

Read the following sections:

- **/etc/hosts** on page 507 of Sobell
- "Introduction to OpenSSH" on page 686 of Sobell
- "Running the ssh, scp, and sftp OpenSSH Clients" on page 689 of Sobell

Procedure

All the labs that work with the classroom server depend on step 1 (below)

tip The first step of this lab puts the hostname **class** in the **/etc/hosts** file, allowing you to refer to the classroom server as **class** instead of having to use its IP address. All the labs that have you work with the classroom server use the hostname **class**; these labs will not work if you do not follow step 1 to set up the **/etc/hosts** file on the local system.

The username **rose** and the password **fit714tree** for the classroom server are set up when the instructor sets up the classroom server (Lab 46, step 7 on page 154 of this Lab Manual).

1. Set up the **/etc/hosts** file so you can refer to the classroom server using the hostname **class** rather than its IP address. (See the tip on page 155 of this Lab Manual.)

   a. Working with **root** privileges, use a text editor to open the **/etc/hosts** file.

```
$ su -c 'vi /etc/hosts'
```

b. Add a line that identifies the classroom server to the bottom of the **hosts** file (Sobell, page 507). Replace *classroom-server-IP* with the IP address the instructor assigned to the classroom server when it was set up.

```
$ cat /etc/hosts
...
classroom-server-IP  class
```

c. Write out the file and exit from the text editor.

Give the command ZZ.

Prompts in this Lab Manual

tip By default, Fedora sets up bash to display a prompt that includes the name of the user you are working as and the name of the working directory, enclosed within square brackets. When you first log in as **student**, the prompt is

```
[student@linux ~]$
```

While you are working as **student**, this Lab Manual shows the prompt simply as **$**.

```
$
```

When you are working as a user other than **student**, the Lab Manual shows the complete prompt. For example, when you are working as Rose on the classroom server, the prompt is

```
[rose@class ~]$
```

And when you are working as Max on the local (student) system, the prompt is

```
[max@linux ~]$
```

2. Connect to the classroom server and issue commands on that system.

a. Use ssh to connect to the classroom server, logging in as Rose (see JumpStart I on page 689 of Sobell).

```
$ ssh rose@class
The authenticity of host 'class (192.168.206.250)' can't be established.
ECDSA key fingerprint is a2:e3:aa:de:cc:30:89:6b:ca:b5:04:d2:91:2d:18:a4.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added 'class,192.168.206.250' (ECDSA) to the list of known hosts.
rose@class's password: fit714tree
Last login: Thu Mar  6 15:41:53 2014
[rose@class ~]$
```

When you connect to an OpenSSH server for the first time, the OpenSSH client (ssh in this case) prompts you to confirm that you are connected to the right system. See "First-time authentication" on page 691 of Sobell for more information. Enter Rose's password for the classroom server when you are prompted for a password. The shell on the classroom server issues a prompt similar to the following:

```
[rose@class ~]$
```

Now when you type commands, you are working on the classroom server.

b. Use the who utility (Sobell, page 254) to see who is logged in on the classroom server. The last column shows where each connection originated. Where did your connection originate? (Remember, you are logged in as Ben.) If several users are logged in as Ben use **who am i** to display only your entry.

```
$ who
$ who am i
```

c. What is the name of the system you are logged in on?

```
$ hostname
```

d. How much space is available in the **/boot** directory on the classroom server? How much space is there in the root (**/**) directory?

```
$ df -h
```

See Sobell, page 804 for more information.

e. How many megabytes of memory are free on the classroom server?

```
$ free -m
```

See Sobell, page 253 for more information.

f. Attempt to run a graphical application such as gnome-calculator while you are logged in on the classroom server. What happens?

The application fails with a **Cannot open display** message.

g. Log off of the classroom server and return to giving commands to the local (student) system by giving an **exit** command or by pressing CONTROL-D.

3. Connect to the classroom server again, this time forwarding X11. See "Forwarding X11" on page 707 of Sobell for more information.

a. Connect to the classroom server again, this time using ssh with the **–X** option.

```
$ ssh -X rose@class
```

b. Now what happens when you run gnome-calculator while you are logged in on the classroom server?

Although the application displays an error message, the local (student) system displays the output of gnome-calculator (a calculator) that is being run on the remote system (the classroom server). See the tip on page 459 of Sobell for more information.

c. Try running a graphical administrative program such as gnome-disks while you are logged in on the classroom server.

d. Log off of the classroom server.

4. When you give ssh a command following the name of the remote system, it executes that command on the remote system. Run gnome-disks on the classroom server without logging in on the classroom server. You may need to press CONTROL-C to get the local shell to display a prompt when you are finished.

```
$ ssh -X rose@class gnome-disks
```

DELIVERABLES

Practice using ssh to connect to a remote system.

Lab 19: Adding a Local, Text-Only Printer (20–30 minutes)

Objectives

You will configure the system to print to a nonexistent printer on the local system.

Setup

- A Fedora 19 installation that includes a nonprivileged user named **student**. You must know the **root** password
- You do *not* need to have a printer attached to the local system; this lab sets up a fictitious printer

Reading

Read the following section:

- "JumpStart II: Setting Up a Local or Remote Printer" on page 560 of Sobell

Procedure

This lab assumes that no printer is attached to the local system. If there is an attached printer, follow the instructions in the adjacent tip to temporarily remove it from the local system.

Turn off and remove a printer on the local system

tip Fedora automatically configures and installs a printer attached to a system. This lab requires that no printer be attached to the local system. Turn off the printer and use the system-config-printer (Print Settings window; Figure 13-1 on page 558 of Sobell) utility to remove it from the system. When you are finished with this lab, remove the printer you set up for the lab. When you turn on the local printer, Fedora will once again set it up.

This lab sets up a fake printer you can experiment with

tip This lab sets up a printer that is connected to a serial port. The system never checks that an actual printer exists, it just sends output to the port. You can experiment with this printer by sending jobs to it and by enabling and disabling it. When you are done experimenting, remove the printer.

1. Display the Print Settings window (system-config-printer; Figure 13-1 on page 558 of Sobell) and use it to set up a local (fake) printer.

   a. Give the command **system-config-printer** from the command line or press the SUPER (Windows) key to display the Search text box and enter **Print Settings**.

Work with root privileges

tip During this lab, system-config-printer will display the Authentication Required window several times; enter the **root** password each time. Alternately, you can run system-config-printer while working with **root** privileges and system-config-printer will not ask you to authenticate.

b. Click **Add** to configure a new printer. The system-config-printer utility displays the Select Device screen of the New Printer window (Sobell, Figure 13-3 on page 561).

c. When system-config-printer asks if you want to adjust the firewall, click **Adjust Firewall**. This action allows other systems on the local network to see printers you set up on the local system.

d. Click **Serial Port #1**, leave the default settings, and click **Forward**. After a moment, system-config-printer utility displays the first Choose Driver screen of the New Printer window (Sobell, Figure 13-5 on page 563).

e. Because no printer is attached to the system, Fedora will not find any drivers. Click **Generic** to select a generic driver from the printer database. Click **Forward**; system-config-printer displays the second Choose Driver screen of the New Printer window (Sobell, Figure 13-6 on page 563).

f. Click **text-only** in the column on the left. Click **Forward**; system-config-printer displays the Describe Printer screen of the New Printer window (Sobell, Figure 13-7 on page 564).

- Replace the text in the text box labeled **Printer Name** with the word **printer** followed by a period and your initials (e.g., **printer.STG**).
- Replace the text in the text box labeled **Description** with a description of the printer (e.g., **my generic printer**).
- Enter some text in the text box labeled **Location** if you like.

Click **Apply**; system-config-printer asks if you want to print a test page. Click **Cancel**.

The tick in a green circle over the printer indicates it is the default printer (Sobell, Figure 13-1 on page 558). In a setup with a single printer you cannot specify a default printer: The only printer is the default printer.

g. Close the Print Settings window.

2. Work with the printer from the command line. Open a terminal emulator window.

a. Display information about the printer.

```
$ lpstat -a
$ lpq
```

b. Use the command line to print the **/etc/hosts** file to the default printer.

```
$ lpr /etc/hosts
```

c. The output will “print” too quickly to view in the queue. Disable the queue and print the **/etc/hosts** file again. Does the print job you just submitted appear in the print queue?

```
$ su -c 'cupsdisable printer-name'
$ lpr /etc/hosts
$ lpq
```

d. Next, send **/etc/sysconfig/network** to the default printer, display the print queue, delete the job, and display the print queue again. (*Hint:* You will need to know the print job number.)

```
$ lpr /etc/sysconfig/network
$ lpq
$ lprm job-number
$ lpq
```

e. Reenable the print queue and check that the remaining job was released.

```
$ su -c 'cupsenable printer-name'
$ lpq
```

3. Use system-config-printer to remove the fake printer from the system.

Deliverables

Practice setting up and sending files to a local printer.

LAB 20: USING CUPS TO CONNECT TO A REMOTE PRINTER (10–15 MINUTES)

OBJECTIVES

In this lab you will define a local printer queue that prints to a remote IPP printer.

SETUP

- A Fedora 19 installation that includes a nonprivileged user named **student**. You must know the **root** password.
- Access to the classroom server. See the tip on page 73 of this Lab Manual.

READING

Read the following section:

- "JumpStart II: Setting Up a Local or Remote Printer" on page 560 of Sobell

PROCEDURE

1. Give the command **system-config-printer** to display the Print Settings window. To avoid repeated requests for authentication, see the tip on page 78 of this Lab Manual.
2. In the Print Settings window, click **Add** to create a new printer.
3. Click to expand the Network Printer section and click **Find Network Printer**; system-config-printer displays a text box labeled **Host** on the right side of the window.
4. Enter **class** in the text box labeled **Host** and then click **Find**. The system-config-printer utility displays the address of the printer in the text box (e.g., **ipp://192.168.206.250:631/printers/fake**). Click **Forward**.
5. On the next screen, specify the name **ghost**, the description **fake**, and the location **classroom server**. Click **Forward**. Printing a test page will fail because the test page is not text-only. Close the Print Settings window.
6. Test the printer by sending a file (e.g., **/etc/hosts**) to the printer from the command line.

```
$ lpr -P ghost /etc/hosts
```

DELIVERABLES

A local printer queue named **ghost** for the printer on the classroom server named **fake**.

Lab 21: Displaying Network Configuration Information (5–10 minutes)

Objectives

Display network configuration settings for the local system using both the GUI and the CLI.

Setup

- A Fedora 19 installation that includes a nonprivileged user named **student**.

Reading

Read the following sections:

- "Introduction to Networking" on page 280 of Sobell
- "Types of Networks and How They Work" on page 282 of Sobell
- "NetworkManager: Configures Network Connections" on page 637 of Sobell

Procedure

1. Work with the Network window to display network information.
   a. Click the NetworkManager applet on the Top panel (Sobell, Figure 16-2 on page 638) and select **Network Settings**; GNOME displays the Network window (Sobell, Figure 16-4 on page 639).
   b. Click the gear at the lower-right corner of the window to display the Wired window. If the local system is not connected to a wired network a different window will be displayed; proceed. Using the tabs labeled **Details** and **Identity**, record the following information.
      - Interface name:____________________________
      - IP Address:____________________________
      - Default Route:____________________________
      - Primary DNS server:____________________________
   c. Close the Wired window and then close the Network window.
2. Use the CLI to display network connection information. Open a terminal emulator window.
   a. Use the ip utility to display information about all network connections. The first connection, named **lo**, is the loopback interface.

```
$ ip addr
```

b. Use the ip utility to display information about the primary external network connection. Replace *interface-name* with the interface name displayed in step 1b (it appears at the start of the second entry).

```
$ ip addr show interface-name
```

c. Use the ip utility to display the address of the default route.

```
$ ip route
```

d. Display the name of the local host (Sobell, page 219).

```
$ hostname
```

e. Display the contents of the **/etc/hosts** file (Sobell, page 507).

```
$ cat /etc/hosts
```

f. Display the contents of the **/etc/resolv.conf** file (Sobell, page 510).

```
$ cat /etc/resolv.conf
```

Deliverables

Practice displaying information about the network configuration of the local system.

LAB 22: CONFIGURING THE FIREWALL USING THE firewall-config UTILITY (10–15 MINUTES)

OBJECTIVES

In this lab you will use the graphical firewall-config utility to configure the firewall on the local system.

SETUP

- A Fedora 19 installation that includes a nonprivileged user named **student**. You must know the **root** password.
- Access to the classroom server. See the tip on page 73 of this Lab Manual.

READING

Read the following sections:

- "The firewalld Service" on page 898 of Sobell
- "JumpStart: Building a Firewall Using firewall-config" on page 900 of Sobell
- "firewall-config: The Firewall Configuration Window" on page 902 of Sobell

PROCEDURE

1. This lab tests the firewall of the local system by using ssh to connect to the classroom server and then connecting back to the local (student) system using ssh.

   a. Give an **ip addr** command and make note of the IP address of the local (student) system. Use this address in place of *myIP* throughout this lab.

   b. Use ssh to connect to the classroom server, logging in as Rose (see JumpStart I on page 689 of Sobell).

   ```
   $ ssh rose@class
   ```

   c. Use ssh to connect from the classroom server back to the local (student) system. Connect as the user named **ben**. Substitute the IP address of the local (student) system for ***myIP*** in the following command.

   ```
   $ ssh ben@myIP
   ```

   As installed, Fedora will accept an ssh connection from a remote system. The first time you connect from the classroom server back to the local (student) system, you will have to go through first-time authentication (Sobell, page 691).

d. When you can connect back to the local (student) system, you know the local firewall did not block the connection.

e. Give an **exit** command to disconnect from the local (student) system and resume working as Rose on the classroom server.

```
[ben@linux ~]$ exit
logout
Connection to 192.168.206.146 closed.
[rose@class ~]$
```

f. Give another **exit** command to disconnect from the classroom server and resume working as **student** on the local (student) system.

```
[rose@class ~]$ exit
logout
Connection to class closed.
[student@linux ~]$
```

2. Work with the Firewall Configuration window (Sobell, Figure 25-1 on page 901) on the local (student) system to put the firewall in panic mode, take it out of panic mode, and open the firewall to allow HTTP (Web) traffic as would be required for an Apache server.

a. Make sure you are working on the local (student) system.

```
$ hostname
linux.example.com
```

b. Open the Firewall Configuration window by giving the command **firewall-config** from the command line or entering **Firewall** in the Search text box. Enter the **root** password and click Authenticate when firewall-config displays the Authentication Required dialog box. Alternately, you can run firewall-config as a privileged user and it will not ask you to authenticate.

c. Set up the Firewall Configuration window as follows:

- Select **Permanent** in the drop-down list labeled **Configuration**.
- Select **Zones** from the main tabs.
- Select **public** from the list on the left labeled **Zone**.
- Select **Services** from the secondary tabs.

d. In its permanent configuration, will the firewall allow inbound ssh traffic as is necessary for an OpenSSH server? How do you know?

Yes, the tick in the check box labeled **ssh** in the lower-right portion of the window shows that the firewall allows ssh traffic in its permanent configuration.

e. In its runtime configuration, will the firewall allow inbound ssh traffic as is necessary for an OpenSSH server?

   Select **Runtime** in the drop-down list labeled **Configuration**. Yes, the tick in the check box labeled **ssh** in the lower-right portion of the window shows that the firewall allows ssh traffic in its runtime configuration. When you connected back to the local system in step 1 of this lab, you saw that the local firewall did not block an incoming ssh connection.

f. Modify the firewall to prohibit ssh traffic in the *runtime* configuration. Use the test described in step 1 of this lab (you might have to open a second terminal emulator window). Does the firewall permit ssh to connect back to the local (student) system?

   No, ssh displays a **No route to host** error.

g. Reboot the local system as explained in step 1b on page 35 of this Lab Manual. Use the test described in step 1 of this lab again; does the firewall permit ssh to connect back to the local system? Is there a tick in the check box labeled **ssh** that firewall-config displays for the runtime configuration?

   Yes, the firewall reverts to the permanent configuration when you reboot the system. Yes, there is a tick in the check box labeled **ssh.**

h. Modify the firewall to prohibit ssh traffic in the *permanent* configuration. Without rebooting the local system, use the test described in step 1 of this lab; does the firewall permit ssh to connect back to the local system?

   Yes, the firewall permits the connection. The permanent configuration does not alter the runtime configuration, it changes the state the firewall will be in when the system reboots.

3. Experiment with panic mode (Sobell, page 901).

   a. Make sure you are working from the local system without going through the classroom server. If you are not sure, give an **exit** command. Open a new terminal emulator window if necessary.

   b. Make sure there is a tick in the check box labeled **ssh** in the runtime configuration.

   c. From the Firewall Configuration window menu, select **Options⇨Panic Mode**. Alternately, you can put the firewall in panic mode from the command line; see "Panic mode" on page 900 of Sobell. What happens when you use ssh to connect to the classroom server after you put the local system in panic mode?

      Nothing happens, there is no response, no error message.

   d. Now, how can you cause the shell to display a prompt?

      Press CONTROL-C.

   e. Select **Panic Mode** from the menu again to remove the tick adjacent to the menu entry to take the firewall out of panic mode and restore its run-time state.

4. Close the Firewall Configuration window.

Deliverables

Practice setting up the firewall using the graphical firewall-config utility.

Lab 23: Adding Packet Filtering Rules Using firewall-cmd (30–40 minutes)

Objectives

You will use the firewall-cmd CLI utility to configure the **firewalld** firewall.

Setup

- A Fedora 19 installation that includes a nonprivileged user named **student**. You must know the **root** password.
- Access to the classroom server. See the tip on page 73 of this Lab Manual.

Reading

Read the following sections:

- "firewall-cmd: Controlling firewalld from the Command Line" on page 906 of Sobell

Procedure

1. Before you start

   a. Make sure that **firewalld** is running.

   ```
   $ firewall-cmd --state
   ```

   b. Give an **ip addr** command and make note of the IP address of the local (student) system. Use this address in place of *myIP* throughout this lab.

2. Open a port on the local firewall to trust a service in the *runtime* configuration, verify the port is open, and test that the port is open from the classroom server using telnet.

   a. Use firewall-cmd to cause the firewall to accept SMTP packets (TCP port 25; **sendmail**) in the runtime configuration.

   ```
   $ su -c 'firewall-cmd --add-service=smtp'
   ```

   or

   ```
   $ su -c 'firewall-cmd --add-port=25/tcp'
   ```

   b. Use firewall-cmd to verify the change you just made.

   ```
   $ su -c 'firewall-cmd --list-services'
   ```

   or

   ```
   $ su -c 'firewall-cmd --list-ports'
   ```

c. Use ssh to connect to the classroom server as explained in Lab 18, step 2a on page 74 of this Lab Manual.

d. From the classroom server, use telnet to connect to port 25 on the local (student) system.

```
[rose@class ~]$ telnet myIP 25
```

How can you tell if the port is open?

- If the port is open and **sendmail** is running on the local system, **sendmail** displays a welcome message; type **quit** to close the connection.
- If the port is open and **sendmail** is not running on the local system, telnet displays a **Connection refused** error.
- If the port is not open on the local system, telnet displays a **No route to host** error.

Close the connection to the classroom server to resume working on the local system.

3. Open a port on the local firewall to trust a service in the *permanent* configuration, reboot the system, and verify that the port is open.

a. Use firewall-cmd to cause the firewall to accept SMTP packets (TCP port 25; **sendmail**) in the permanent configuration.

```
$ su -c 'firewall-cmd --permanent --add-service=smtp'
```

or

```
$ su -c 'firewall-cmd --permanent --add-port=25/tcp'
```

b. Reboot the local system as explained in step 1b on page 35 of this Lab Manual.

c. Use firewall-cmd to verify that the change you made is in effect in both the running and permanent configurations.

```
$ su -c 'firewall-cmd --list-services'
$ su -c 'firewall-cmd --permanent --list-services'
```

or

```
$ su -c 'firewall-cmd --list-ports'
$ su -c 'firewall-cmd --permanent --list-ports'
```

4. Close the port you opened in the preceding steps in both the running and permanent configurations.

```
$ su -c 'firewall-cmd --remove-service=smtp'
$ su -c 'firewall-cmd --permanent --remove-service=smtp'
```

or

```
$ su -c 'firewall-cmd --remove-port=25/tcp'
$ su -c 'firewall-cmd --permanent --remove-port=25/tcp'
```

5. Set up the local system so that the running firewall blocks ICMP pings from remote systems. See "ping: Tests a Network Connection" on page 305 of Sobell and the discussion of the ICMP filter on page 904 of Sobell. Verify that the setup works by pinging the local system from the classroom server.

The ––permanent option works with ICMP options too

tip As with other firewall-cmd commands, ICMP-related commands work on the running system unless you include the **––permanent** option.

a. Display a list of ICMP types using the firewall-cmd **––get-icmptypes** option. Because this command displays a static list of ICMP types and has no bearing on the firewall nor gives away any information pertinent to system security, you do not need to work with **root** privileges to execute it.

```
$ firewall-cmd --get-icmptypes
```

b. Use the firewall-cmd **––add-icmp-block** option to block ECHO_REQUEST (**echo-request**) packets on the local system. Read the firewall-cmd man page to learn the format of the command.

```
$ su -c 'firewall-cmd --add-icmp-block=echo-request'
```

c. Use the firewall-cmd **––list-icmp-blocks** option to display a list of the types of ICMP packets that the local system will not accept (blocks).

```
$ su -c 'firewall-cmd --list-icmp-blocks'
```

d. Use the firewall-cmd **––remove-icmp-block** option to cause the local system to accept ECHO_REQUEST (**echo-request**) packets.

```
$ su -c 'firewall-cmd --remove-icmp-block=echo-request'
```

Deliverables

An understanding of how to use firewall-cmd to open ports and block ICMP packets.

Lab 24: Installing the Apache (httpd) Web Server (15–20 minutes)

Objectives

In this lab you will set up an Apache (**httpd**) Web server on the local system.

Setup

- A Fedora 19 installation that includes a nonprivileged user named **student**. You must know the **root** password.
- Access to the classroom server. See the tip on page 73 of this Lab Manual.

Reading

Read the following sections:

- "Introduction" on page 932 of Sobell
- "Running an Apache Web Server" on page 935 of Sobell
- "JumpStart: Getting Apache Up and Running" on page 936 of Sobell

Procedure

1. Verify that the **httpd** server is installed on the local system, configure **httpd**, and test **httpd** from the local system.

   a. Check that the **httpd** package is installed; install it if it is not installed.

   ```
   $ rpm -q httpd    # displays the version number if it is installed
   ```

   Alternately, you can just try to install **httpd**; yum will either tell you it is installed or will install it.

   ```
   $ su -c 'yum -y install httpd'
   ```

   b. Read and follow the instructions in the section "Modifying the httpd.conf Configuration File" on page 936 of Sobell. Supply the IP address of the local system in the ServerName directive. Specify **student@localhost** as the email address in the ServerAdmin directive.

   c. Run systemctl to cause the **httpd** service to start each time the system enters multiuser mode and then restart the **httpd** service. Use the systemctl **status** command to make sure the service is running.

   ```
   $ su -c 'systemctl enable httpd.service'
   $ su -c 'systemctl restart httpd.service'
   $ systemctl status httpd.service
   ```

d. Test the Web server from the local system by following the instructions in the section “Testing Apache” on page 937 of Sobell. Does Firefox display the Fedora test page?

Yes, if everything is set up properly, Firefox will display the Fedora test page.

2. Test to see if you can access the local Apache server from a remote system. Open the firewall so that HTTP requests from a remote system can reach the local system and test the setup again.

a. Give an **ip addr** command and make note of the IP address of the local (student) system. Use this address in place of *myIP* throughout this lab.

b. Use ssh with the **–X** option to log in on the classroom server as **rose** and run firefox.

```
$ ssh -X rose@class firefox
```

Alternately, you can omit **firefox** from the preceding command and run it from the command line once you log in.

c. From firefox, which is running on the classroom server and displayed on the local system, contact the Apache server running on the local system. What happens?

Enter **http://*myIP*** in the firefox location bar, where ***myIP*** is the IP address of the local system, and press RETURN. The browser displays an **Unable to connect** message.

d. Open a new terminal emulator window on the local (student) system. From that window add the **http** and **https** services to the firewall on the local system.

```
$ su -c 'firewall-cmd --add-service=http'
$ su -c 'firewall-cmd --add-service=https'
```

or

```
$ su -c 'firewall-cmd --add-port=80/tcp'
$ su -c 'firewall-cmd --add-port=443/tcp'
```

e. Return to the window that is displaying firefox and reload the page. What happens?

The browser displays the Fedora test page.

f. Close the browser (and disconnect from the classroom server).

3. Add content to the HTTP server and verify that a browser displays the new content.

a. Display the location of the document root (pages 934 and 942 of Sobell) by displaying the DocumentRoot directive. Which directory is the document root?

```
$ grep DocumentRoot /etc/httpd/conf/httpd.conf
```

By default, the **/var/www/html** directory is the document root.

b. In the document root directory, work with **root** privileges to create a file named **index.html** that contains the hostname of the local system. Following is a sample file. The **<H2> ... </H2>** tags causes Apache to display the text as a second-level heading.

```
$ cat /var/www/html/index.html
<H2>Welcome to linux.example.com</H2>
```

c. Run firefox on the classroom server and point it toward the local system again. What does the browser display?

The browser displays **Welcome to linux.example.com**.

4. Set up a link to the public FTP directory on the Web page you just created.

a. Install the **vsftpd** FTP server package to create the **/var/ftp/pub** directory (Sobell, page 724). Do not start **vsftpd**.

```
$ su -c 'yum -y install vsftpd'
```

b. Create a symbolic link (Sobell, page 206) to **/var/ftp/pub** in the document root directory.

```
$ su -c 'ln -s /var/ftp/pub /var/www/html/pub'
```

c. Include a link to **pub** in the HTML document so that users can access FTP files while browsing the Web site.

```
$ cat /var/www/html/index.html
<H2>Welcome to linux.example.com</H2>.
See additional files in the <a href="pub">pub</a> directory.
```

5. As in previous steps in this lab, use firefox to test the new setup. What happens when the mouse cursor hovers over the **pub** link? What happens when you click the link?

With the cursor hovering over the link, firefox displays the relative pathname of the link at the lower-left corner of the browser. When you click the link, firefox displays the **pub** directory.

Use elinks to test HTML documents

tip The elinks utility displays a textual representation of a graphical HTML page. After installing the **elinks** package, give the following command

```
$ elinks www.sobell.com
```

The elinks **––dump** option displays a very simple representation of an HTML page that includes a footnote for each link on the page; give the following command

```
$ elinks --dump www.sobell.com | less
```

A numbered list of links appears at the bottom of the output.

6. *Optional:* Use the elinks browser with the **––dump** option to display the link in **index.html**.

   ```
   $ elinks --dump http://localhost
   ```

7. Set up and test an SSL (encrypted) connection as implemented by **mod_ssl** (Sobell, page 970).

   a. Install the **mod_ssl** package.

   ```
   $ su -c 'yum -y install mod_ssl'
   ```

   b. Restart the HTTP service to incorporate **mod_ssl**.

   ```
   $ su -c 'systemctl restart httpd.service'
   ```

   c. As in previous steps in this lab, use firefox to test the new setup. This time connect to **https://localhost** (use **https** in place of **http**). What does the browser display?

   The browser displays a **This Connection is Untrusted** message.

   d. Click **Technical Details** and read the newly displayed information.

   e. Click **I Understand the Risks** and read the newly displayed information.

   f. Click **Add Exception** to display the Add Security Exception window.

   g. Click **Confirm Security Exception** in the Add Security Exception window. What happens?

   The browser displays a secure version of the page.

8. *Challenge:* Read "Using a Self-Signed Certificate for Encryption" on page 970 of Sobell and generate a customized self-signed certificate.

DELIVERABLES

A Web server using the default configuration with a custom index page.

Lab 25: Managing User Content and Private Directories (30–40 minutes)

Objectives

You will set up the Apache Web server to allow users to share a subdirectory of their home directory using Apache.

Setup

- A Fedora 19 installation that includes a nonprivileged user named **student**. You must know the **root** password.

Reading

Read the following section:

- “UserDir” on page 943 of Sobell

Procedure

1. Set up Apache so that Max can share files from the **public_html** subdirectory of his home directory (**~max/public_html**).

   a. Working with **root** privileges, edit the **/etc/httpd/conf.d/userdir.conf** file. Follow the instructions (comments) in the <IfModule> container: Comment out (put a hashmark in front of) the **UserDir disabled** directive and uncomment (remove the hashmark from in front of) the **UserDir public_html** directive. Exit from the editor.

   b. Working as Max, create a **public_html** directory as a subdirectory of Max’s home directory and create a file named **index.html** in that directory. Put the string **the index file belonging to max** in the new file.

   ```
   $ su - max
   [max@linux]$ mkdir ~/public_html
   [max@linux]$ vim ~/public_html/index.html
   ```

   c. Still working as Max, change the group associated with Max’s home directory and his **public_html** directory to **apache**.

   ```
   [max@linux ~]$ su -c 'chgrp apache ~max ~max/public_html'
   ```

   d. Still working as Max, change the permissions on Max’s home directory and his **public_html** directory so that members of the group named **apache** can read and search (execute) those directories. By default, everyone can read the files Max creates.

   ```
   [max@linux ~]$ chmod 750 ~max ~max/public_html
   ```

e. Return to working as the user named **student**.

```
[max@linux ~]$ exit
```

f. Restart the HTTP service.

```
$ su -c 'systemctl restart httpd.service'
```

g. Change the SELinux settings so that it allows HTTP to access users' home directories.

```
$ su -c 'setsebool -P httpd_enable_homedirs on'
```

The preceding command will take a while to complete.

h. Use elinks to test to see if Max is sharing files. What does elinks display?

```
$ elinks --dump http://localhost/~max
```

The elinks utility displays the contents of the **index.html** file in **~max/public_html**: the string **the index file belonging to max.**

2. Max wants to share some family photos with a few relatives. Set up **photos** as a subdirectory of Max's **public_html** directory. Protect the **photos** directory with a username and password (use **maxp** and **photo**) so that Apache only permits users who know the password to browse this directory. In a real-world setup you would choose a more secure password.

a. Working as Max, create the **~/public_html/photos** folder and copy the **/usr/share/pixmaps/faces/sunflower.jpg** file to that folder.

```
$ su - max
[max@linux ~]$ mkdir ~/public_html/photos
[max@linux ~]$ cp /usr/share/pixmaps/faces/sunflower.jpg ~/public_html/photos
```

b. Still working as Max, change the group association and permissions for the **photos** directory as you did for the **public_html** directory (steps 1c and 1d on the previous page).

```
[max@linux ~]$ su -c 'chgrp apache ~max/public_html/photos'
[max@linux ~]$ chmod 750 ~max/public_html/photos
```

c. Create the **~/public_html/photos/.htaccess** file to grant permission to anyone with a username on the local system (Sobell, page 972). Set up the file to store the password in **~/public_html/.htpasswd**.

```
[max@linux ~]$ su -c 'cat ~/public_html/photos/.htaccess'
AuthUserFile /home/max/public_html/.htpasswd
AuthName "Private Photos"
AuthType basic
require valid-user
```

d. Use htpasswd to set up the **~/public_html/.htpasswd** file for the user named **maxp** with a password of **photo**. What does htpasswd put in the **.htpasswd** file? (*Hint:* You can use cat to display this file.)

```
[max@linux ~]$ htpasswd -c ~/public_html/.htpasswd maxp
```

The htpasswd utility puts a username, a colon, and an encrypted password in the **.htpasswd** file.

e. Return to working as the user named **student**. You do not have to restart the HTTP service.

```
[max@linux ~]$ exit
```

f. Use firefox to display the **sunflower** file you copied to Max's **photos** directory.

Open firefox and enter **localhost/~max/photos** in the location bar. Enter **pmax** and **photo** in response to the username and password prompt. With firefox displaying the contents of the **photos** directory, click **sunflower.jpg**.

Deliverables

A Web server offering users a place to offer personal content including password-protected content.

Lab 26: Transferring Files Securely Using scp and rsync (20–30 minutes)

Objectives

In this lab you will use scp and rsync to securely transfer files between systems and automate these procedures using OpenSSH keys.

Setup

- A Fedora 19 installation that includes nonprivileged users named **student** and **rose**. You must know the **root** password.
- Access to the classroom server. See the tip on page 73 of this Lab Manual.

Reading

Read the following sections:

- "Introduction to OpenSSH" on page 686 of Sobell
- "How OpenSSH Works" on page 687 of Sobell
- "Running the ssh, scp, and sftp OpenSSH Clients" on page 689 of Sobell
- "JumpStart I: Using ssh and scp to Connect to an OpenSSH Server" on page 689 of Sobell
- "ssh: Logs in or Executes Commands on a Remote System" on page 693 of Sobell
- "scp: Copies Files to and from a Remote System" on page 695 of Sobell
- "rsync is much more versatile than scp" on page 696 of Sobell
- "Authorized Keys: Automatic Login" on page 700 of Sobell

Procedure

1. Transfer files securely using scp and rsync.

   a. Use scp to copy the **/etc/hosts** file from the local system to Rose's working directory on the classroom server. Specify the username **rose** when copying the file.

   ```
   $ scp /etc/hosts rose@class:
   ```

   b. Use scp to copy the **/etc/hosts** file from the local system to the **/tmp** directory on the classroom server. Specify the username **rose** when copying the file. Name the new file **hosts-*XXX*** (replace ***XXX*** with your initials).

   ```
   $ scp /etc/hosts rose@class:/tmp/hosts-XXX
   ```

c. Use ssh on a single command line to display the contents of the copied file from step b.

```
$ ssh rose@class cat /tmp/hosts-XXX
```

d. Repeat step b using rsync in place of scp; add the number **2** to the end of the name of the copied file (**hosts-*XXX*2**).

```
$ rsync /etc/hosts rose@class:/tmp/hosts-XXX2
```

e. Use rsync to copy the **/etc/hosts** file on the classroom server to your home directory on the local system.

```
$ rsync rose@class:/etc/hosts ~
```

f. Use rsync to copy the **mine** script from step 7 on page 70 of this Lab Manual to Rose's home directory on the classroom server. Append your initials to the name of the copied file

```
$ rsync ~/mine rose@class:mine.XXX
```

You do not need to quote ssh arguments

tip Unlike some utilities, you do not need to quote commands that you pass to ssh to execute on a remote system. For example, the following command displays a long listing of the **warts** file in Ben's home directory on the system named **mash**:

```
$ ssh ben@mash ls -l warts
```

g. Compare the file permissions on the source (on the local system) **mine** file and destination **mine** file (on the classroom server). Are they the same? Is the destination file executable?

```
$ ls -l mine
$ ssh rose@class ls -l mine.XXX
```

Both rsync and scp copy file permissions when they copy a file. The permissions are the same. If the source file is executable (it should be) then the destination file is executable. Because Rose copied the file, she owns the copy.

h. Run **mine.*name*** on the remote system.

```
$ ssh rose@class ./mine.XXX
```

2. Set up and test authorized key authentication for Rose on the classroom server.

a. Start working as Rose.

```
$ su - rose
Password: fit714tree
[rose@linux ~]$
```

b. Because you are working as Rose, you do not have to specify a username when you use ssh to connect to Rose's account on the classroom server. Use ssh to connect as Rose to the classroom server and then disconnect so that you are once again working on the local (student) system. If this is the first time you are connecting to the classroom server while working as Rose, you will go through first-time authentication.

```
$ ssh class
The authenticity of host 'class (192.168.206.250)' can't be established.
ECDSA key fingerprint is a2:e3:aa:de:cc:30:89:6b:ca:b5:04:d2:91:2d:18:a4.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added 'class,192.168.206.250' (ECDSA) to the list of known hosts.
rose@class's password: fit714tree
[rose@class ~]$ exit
logout
Connection to class closed.
[rose@linux ~]$
```

c. Use ssh-keygen (Sobell, page 701) to create an ssh key pair with an empty passphrase for **student** on the local system. Verify that the **id_rsa** and **id_rsa.pub** files exist in Rose's **~/.ssh** directory.

```
[rose@linux ~]$ ssh-keygen
[rose@linux ~]$ ls ~/.ssh
```

d. Use ssh-copy-id (Sobell, page 702) to copy the public key in the account of the remote user. You can ignore INFO messages about keys. These messages are the result of users from several remote systems (many student systems) creating authorized keys for the same user.

```
[rose@linux ~]$ ssh-copy-id class
```

e. Test connectivity by using ssh to run the hostname utility on the classroom server. No password should be required.

```
[rose@linux ~]$ ssh class hostname
```

3. By default, rsync uses OpenSSH to authenticate and transfer files. The authorized key you set up for Rose on the classroom server will make it so that rsync does not request a password.

a. Still working as Rose, use rsync to transfer the **/etc/passwd** file from the classroom server to a file named **passwd.class** in your home directory on the local system. Verify that the file was copied.

```
[rose@linux ~]$ rsync class:/etc/passwd passwd.class
[rose@linux ~]$ ls passwd.class
```

b. Display the last line of the file you just transferred. Which user does the line define?

```
[rose@linux ~]$ tail -1 passwd.class
```

The line defines Rose.

4. Return to working as **student**.

```
[rose@linux ~]$ exit
logout
[student@linux ~]$
```

DELIVERABLES

Practice using ssh to access a remote system using passwords and authorized keys, and practice transferring files securely using both scp and rsync.

LAB 27: CUSTOMIZING USERS AND THE SYSTEM USING SCRIPTS (60–90 MINUTES)

OBJECTIVES

You will modify and create scripts to manage the user environment and automate system administration tasks.

SETUP

- A Fedora 19 installation that includes nonprivileged users named **student** and **casey**. You must know the **root** password.

READING

Read the following sections:

- "Startup Files" on page 329 of Sobell
- "Writing and Executing a Shell Script" on page 336 of Sobell
- "PATH: Where the Shell Looks for Programs" on page 359 of Sobell
- "Aliases" on page 392 of Sobell

PROCEDURE

1. Customize Casey's shell environment.

   a. Use su to start working as the user named **casey**.

   ```
   $ su - casey
   Password:
   [casey@linux ~]$
   ```

   b. Before you start, preserve variable and alias definitions in Casey's **.bashrc** and **.bash_profile** startup files by making backup copies of these files.

   ```
   [casey@linux ~]$ cp .bashrc bashrc.0
   [casey@linux ~]$ cp .bash_profile bash_profile.0
   ```

   c. Members of the **staff** group will use the **/staff/bin** shared directory that will holds scripts. Ensure that this directory is a part of Casey's **PATH** (Sobell, page 359) each time she logs in on the system.

   Use a text editor to edit the **/home/casey/.bash_profile** file and append **:/staff/bin** to the string that is assigned to the **PATH** variable:

   ```
   PATH=$PATH:$HOME/.local/bin:$HOME/bin:/staff/bin
   ```

d. Casey has a habit of removing files by using overly broad filename generation wildcards. Add an alias so that rm prompts her each time she uses it to remove a file.

   Use an editor to add the following line to the end of the **/home/casey/.bashrc** file.

   ```
   alias rm='rm -i'
   ```

2. Write a script that generates a system activity report that includes disk and memory usage.

   a. Use vim to edit the file that will hold the script. Name the file **report.sh**.

   ```
   $ vim report.sh
   ```

   b. The first line of the script should specify that the script will run under the bash shell (**#!**; Sobell, page 338). The second line should be a comment that describes the purpose of the script (**#**; Sobell, page 340).

   ```
   #!/bin/bash
   # This script generates a system activity report.
   ```

   c. Start the report with a today's date.

   ```
   date
   ```

   d. Be sure the report contains the hostname of the system.

   ```
   hostname
   ```

   e. Report on the amount of free disk space on the system.

   ```
   df -h
   ```

   f. Report on the amount of memory used by the system.

   ```
   free -hm
   ```

   g. If the **root** user is logged in on the system, display the name of the terminal device that account is using.

   ```
   who | grep root
   ```

   h. You may want to clean up the output with some blank lines or descriptions.

   i. Save the script. It might look something like this:

   ```
   #!/bin/bash
   # This script generates a system activity report.
   date
   hostname
   echo
   df -h
   free -hm
   ```

```
echo
echo "root is logged on the following terminals"
who | grep root
```

j. Make the new script executable (Sobell, page 337).

```
$ chmod 755 report.sh
```

k. Create a subdirectory of your home directory named **bin**. You can refer to this directory as **~/bin** (Sobell, page 182). This directory is included in your **PATH** variable by default. Move the shell script to the **~/bin** directory.

```
$ mkdir ~/bin
$ mv report.sh ~/bin
```

l. Test the script. You can specify a simple filename to call the script because the file is in a directory that is in your **PATH**.

```
$ report.sh
```

3. Resume working as **student**.

Deliverables

A customized environment for Casey to work in and a shell script that displays a system activity report.

Lab 28: The systemd init Daemon (30 minutes)

Objectives

This lab explains how to use **systemd** to bring the system to different target units (runlevels) and start and stop services (daemons).

Setup

- A Fedora 19 installation that includes a nonprivileged user named **student**. You must know the **root** password.

Reading

Read the following sections:

- "Using a Virtual Console" on page 121 of Sobell
- "The systemd init Daemon" on page 438 of Sobell
- "Service Units and Target Units" on page 439 of Sobell
- "Runlevels" on page 440 of Sobell
- "Wants and Requires" on page 440 of Sobell
- "Determining Whether systemd Runs a Daemon Natively" on page 443 of Sobell
- "Setting and Changing Runlevels" on page 444 of Sobell
- "Configuring Daemons (Services)" on page 445 of Sobell
- "Runlevels" on page 449 of Sobell

Procedure

The persistent state of a service (daemon) is the state it will be in (enabled or disabled) when the system is booted. The current state of a service is active (running) or inactive (not running). Changing the persistent state of a service does not affect its current state and vice-versa. See "Configuring Daemons (Services)" on page 445 of Sobell for more information.

1. Open a terminal emulator window (Lab Manual, page 15).
2. Display and change the current and persistent state of the **sshd** service.

   a. Give the following command to display the persistent state of the **sshd** service.

   ```
   $ systemctl is-enabled sshd.service
   enabled
   ```

b. Give a command to cause the **sshd** service not to start when the system is booted. Do you need to work with **root** privileges? Why?

```
$ su -c 'systemctl disable sshd.service'
rm '/etc/systemd/system/multi-user.target.wants/sshd.service'
```

Yes, you need to work with **root** privileges because you are making a global change to the system.

c. Verify that the **sshd** service is disabled.

```
$ systemctl is-enabled sshd.service
disabled
```

d. Give the following command to display the current state of the **sshd** service.

```
$ systemctl is-active sshd.service
active
```

e. Give a command that will stop the **sshd** service from running immediately and then verify that it is inactive.

```
$ su -c 'systemctl stop sshd.service'
$ systemctl is-active sshd.service
inactive
```

f. Restart the **sshd** service.

```
$ su -c 'systemctl start sshd.service'
```

g. Display the status of the **sshd** service using a **systemctl status** command. How long ago was the service started? What is the status of the service?

```
$ systemctl status sshd.service
```

If you are following this lab, it was started less than a minute ago. The status is **active (running)**.

3. By default, Fedora boots to multi-user.target, also called multiuser graphical mode or runlevel 5. Give the following command to display the current system target (runlevel):

```
$ systemctl list-units --type=target | egrep 'rescue|multi-user|graphical'
graphical.target    loaded active active Graphical Interface
multi-user.target   loaded active active Multi-User System
```

This command sends the output of systemctl through a pipeline to egrep (extended grep; grep cannot search for multiple stings at one time), which searches for each of the strings in its argument. The output shows the system is running in graphical.target. It also shows multi-user.target because graphical.target is dependent on multi-user.target: graphical.target cannot run if multi-user.target is not running.

4. Change the current target unit to rescue.target (also called single-user mode, runlevel 1, and rescue mode).

   a. Working with **root** privileges, give the following command to change to rescue.target. In a multiuser or graphical environment you would want to make sure no one else was logged in before giving this command. See "Changing the Current Runlevel" on page 445 of Sobell for more information.

   ```
   $ su -c 'systemctl isolate rescue.target'
   ```

   After you give this command, the system shuts down the GUI, displays a textual CLI, and displays the following prompt:

   ```
   Give root password for maintenance
   (or type Control-D to continue)
   ```

   b. If you press CONTROL-D at this point, the system reboots to graphical.target. Instead, type the **root** password and press RETURN; the system enters rescue.target (single-user mode) and displays a root prompt:

   ```
   [root@linux ~]#
   ```

   You are working with **root** privileges in rescue.target.

   c. Give the following command to confirm the system is running in rescue.target (rescue mode):

```
[root@linux ~]# systemctl list-units --type=target | egrep 'rescue|multi-user|graphical
rescue.target      loaded active active Rescue Mode
```

5. Change the current target unit to graphical.target (previously called runlevel 5).

   a. Working with **root** privileges (you automatically work with **root** privileges while the system is in single-user mode), give the following command to change to graphical.target.

   ```
   [root@linux ~]# systemctl isolate graphical.target
   ```

   The system boots to graphical mode and displays the Login screen.

   b. Log in as Sammy Student.

   c. Open a terminal emulator window and give the following command to display the target the system is running in.

```
$ systemctl list-units --type=target | egrep 'rescue|multi-user|graphical'
graphical.target    loaded active active Graphical Interface
multi-user.target   loaded active active Multi-User System
```

6. Change the current target unit to multi-user.target (previously called multi-user mode or runlevel 3).

a. Working with **root** privileges, give the following command to change to multi-user.target.

```
$ su -c 'systemctl isolate multi-user.target'
```

b. The system shuts down the GUI and displays a textual CLI that shows a login prompt. Log in as **root** (provide the **root** password) and give the following command to display the target the system is running in.

```
[root@linux ~]# systemctl list-units --type=target | egrep 'rescue|multi-user|graphical'
multi-user.target   loaded active active Multi-User System
```

7. Determine the persistent target (the target the system boots to by default), reboot the system, and verify the new target.

a. By default, Fedora sets the persistent target to graphical.target. The **/etc/systemd/system/default-target** file is a link to the file that defines the default persistent target. Which is the persistent target?

```
[root@linux ~]# ls -l /etc/systemd/system/default.target
lrwxrwxrwx. 1 root root 36 Jan  9 12:25 /etc/systemd/system/default.target ->
/lib/systemd/system/graphical.target
```

b. Give the follwing command to reboot the system. Because you are logged in as **root**, you are working with **root** privileges and do not need to use su.

```
[root@linux ~]# reboot
```

The system reboots and displays the GUI Login screen.

c. Log in as Sammy Student, open a terminal emulator window, and display the name of the target the system is running in.

```
$ systemctl list-units --type=target | egrep 'rescue|multi-user|graphical'
graphical.target    loaded active active Graphical Interface
multi-user.target   loaded active active Multi-User System
```

DELIVERABLES

Practice starting and stopping services and bringing the system to different target units (runlevels).

Lab 29: Working with SELinux (15–20 minutes)

Objectives

In this lab you will explore ways to work with SELinux.

Setup

- A Fedora 19 installation that includes a nonprivileged user named **student**. You must know the **root** password.

Reading

Read the following section:

- "SELinux" on page 472 of Sobell

Introduction

Fedora and Red Hat Enterprise Linux default installations enable SELinux in Enforcing mode and apply a policy that uses type enforcement. This setup can protect your system from damage when an intruder gains access to the system; however, it can also get in the way of adding additional software to the system until you have a stronger understanding of the contexts, Booleans, and policy involved in configuring SELinux.

Procedure

1. Display information about the SELinux configuration.

   a. Give the command **sestatus**.What is the status of SELinux? What is its current mode? What is the mode from the **config** file? Are they the same?

   ```
   $ sestatus
   SELinux status:                 enabled
   ...
   Current mode:                   enforcing
   Mode from config file:          enforcing
   ...
   ```

   The current mode and the mode from the **config** file are the same (enforcing).

   b. Display the **/etc/selinux/config** file. All the lines that start with hash marks (#) are comments. What do the two noncomment lines tell you? When do these settings apply?

   ```
   $ cat /etc/selinux/config file
   ...
   SELINUX=enforcing
   ```

```
...
SELINUXTYPE=targeted
```

The lines tell you that the system boots to SELinux targeted enforcing mode. The settings in the **config** file take effect immediately after the system boots.

2. Use setenforce to put SELinux in permissive mode. Now what does sestatus show about the current mode and the mode from the **config** file? Which mode is in effect? Which mode would be in effect if you rebooted the system?

   ```
   $ su -c 'setenforce permissive'
   $ sestatus
   ...
   Current mode:                   permissive
   Mode from config file:          enforcing
   ...
   ```

   The current mode and the mode from the **config** file are the different; permissive mode is in effect. The settings from the **config** file apply when the system is rebooted; enforcing mode would be in effect once the system was rebooted.

3. Edit the **/etc/selinux/config** file and change the system default mode to **permissive**. Now which mode would be in effect if you rebooted the system?

   ```
   $ su -c 'vim /etc/selinux/config'
   ```

   Set **SELINUX=permissive**. The settings from the **config** file apply when the system is rebooted; permissive mode would be in effect once the system was rebooted.

4. Use sestatus to change the mode back to enforcing.

   ```
   $ su -c 'setenforce enforcing'
   ```

5. *Optional:* Set up SELinux so that it allows users to share their home directories using Samba.

   a. Use getsebool with the **–a** option to list SELinux Booleans that pertain to Samba. Which of these Booleans pertain to home directories?

   ```
   $ getsebool -a | grep -i samba
   ...
   samba_create_home_dirs --> off
   use_samba_home_dirs --> off
   ...
   ```

   b. Read the setsebool man page. Use setsebool to set to **on** the SELinux Booleans that pertain to Samba and home directories. Make sure the Booleans will retain their values after the system is rebooted. These commands take a little while to run.

```
$ su -c 'setsebool -P samba_create_home_dirs on'
$ su -c 'setsebool -P use_samba_home_dirs on'
```

c. Verify the changes you made to the SELinux Booleans.

```
$ getsebool -a | grep -i samba
...
samba_create_home_dirs --> on
use_samba_home_dirs --> on
...
```

DELIVERABLES

Practice working with SELinux.

LAB 30: GAINING root PRIVILEGES USING sudo
(15–20 MINUTES)

OBJECTIVES

In this lab you will configure sudo to allow specific users to issue specific administrative commands using **root** privileges without those users needing to know the **root** password.

SETUP

- A Fedora 19 installation that includes the nonprivileged users named **student**, **max**, and **casey**. You must know the **root** password.

READING

Read the following sections:

- "Using sudo to Gain root Privileges" on page 428 of Sobell
- "sudoers: Configuring sudo" on page 433 of Sobell

PROCEDURE

1. Working with **root** privileges, use visudo to modify the **/etc/sudoers** file (Sobell, page 433).

   ```
   $ su -c visudo
   ```

   Add a line to the end of the file that grants the user named **student** full **root** privileges (while running sudo).

   ```
   student ALL=(ALL) ALL
   ```

2. Working as **student**, try to display the end of the **/var/log/messages** file without using sudo. Next, display the same file using sudo; you will need to supply *your* password (not the **root** password).

   ```
   $ tail /var/log/messages
   $ sudo tail /var/log/messages
   ```

3. Working as **student**, use sudo to gain **root** privileges and use visudo to modify the **/etc/sudoers** file. Did you have to enter your password? Why or why not?

   ```
   $ sudo visudo
   ```

   No, you probably did not need to enter your password. The sudo utility maintains a timestamp for the last time a user called sudo. If you call sudo within five minutes of the last time you called it, sudo does not require a timestamp. See “Timestamp” on page 429 of Sobell for more information.

Add lines to the end of the **sudoers** file that

a. Add a user alias named **ADMIN** that includes **max** and **casey**.

```
User_Alias ADMIN = max, casey
```

b. Add a command alias named **MEDIA** that includes **mount** and **umount**.

```
Cmnd_Alias MEDIA = /bin/mount, /bin/umount
```

c. Allow **ADMIN** users and **ben** to run the **MEDIA** commands.

```
ADMIN, ben ALL=(ALL) MEDIA
```

4. Working as **ben**, **casey**, and then **max**, test the modified **sudoers** file as follows

a. First, change your effective UID to **ben** and verify your effective UID. After testing as **ben**, repeat each of the steps in this section for **casey** and **max**.

```
$ su - ben
[ben@linux ~]$ whoami
```

b. Use df to verify that the **/boot** filesystem is mounted.

```
[ben@linux ~]$ df /boot
```

c. Attempt to unmount **/boot** without using sudo.

```
[ben@linux ~]$ umount /boot
```

d. Use sudo to unmount **/boot**.

```
[ben@linux ~]$ sudo umount /boot
```

e. Use df to verify that the **/boot** filesystem is no longer mounted.

```
[ben@linux ~]$ df /boot
```

When **/boot** is not mounted, this command displays the root partition (**/**).

f. Use sudo to mount **/boot**.

```
[ben@linux ~]$ sudo mount /boot
```

g. Ensure that **ben**, **casey**, and **max** cannot run other commands using sudo. For example, while working as **ben**, use sudo to attempt to display the **/var/log/messages** file.

```
[ben@linux ~]$ sudo less /var/log/messages
Sorry, user ben is not allowed to execute '/bin/less
/var/log/messages' as root on linux.example.com.
```

h. Return to working with, and verify you are working with, an effective UID of **student**.

```
[ben@linux ~]$ exit
$ whoami
```

Deliverables

A **sudoers** file granting the user **student** full administration privileges and granting select privileges to other users.

LAB 31: INSTALLING SOFTWARE USING rpm (10–15 MINUTES)

OBJECTIVES

In this lab you will install software packages and their dependencies using the rpm utility.

SETUP

- A Fedora 19 installation that includes a nonprivileged user named **student**. You must know the **root** password.

READING

Read the following sections:

- "RPM: The RPM Package Manager" on page 546 of Sobell
- "Querying Packages and Files" on page 547 of Sobell
- "Installing, Upgrading, and Removing Packages" on page 548 of Sobell

Finding software package files on the Install DVD or ISO image file

tip Once you mount the Fedora install DVD or DVD ISO image file and determine where it is mounted as explained in step 1 and step 2 of this lab, you can find the software package (***.rpm**) files. All packages are located in subdirectories of the **Packages** directory, which is a subdirectory of the mount point for the install DVD or DVD ISO image file. Within the **Packages** directory are up to 26 directories, each named for a letter of the alphabet. Software packages are listed in the directory named by the first letter of the name of the package. For example, if df shows the mount point for the Install DVD or DVD ISO image file as

```
$ df | grep Fedora
... /run/media/student/Fedora 19 i386
```

the following command displays the contents of the **Packages** directory

```
$ ls /run/media/student/Fedora*/Packages
a  b  c  d  e  f  g  h  i  j  k  l  m  n  o  p  q  r  s  t ...
```

and the next command displays the names of all package files that begin with **samba**

```
$ ls /run/media/student/Fedora*/Packages/s/samba*
/run/media/student/Fedora 19 i386/Packages/s/samba-client-4.0.6-
3.fc19.i686.rpm
/run/media/student/Fedora 19 i386/Packages/s/samba-common-4.0.6-
3.fc19.i686.rpm
/run/media/student/Fedora 19 i386/Packages/s/samba-libs-4.0.6-
3.fc19.i686.rpm
```

Procedure

1. Mount the Fedora Install DVD or DVD ISO image file as explained in Lab 45 on page 151 of this Lab Manual. Do not set it up as a repository as the subsequent steps of Lab 45 describe. If it is already mounted, continue.
2. Open a terminal emulator window and then determine where the DVD or ISO image file is mounted as explained in step 3 of Lab 45 on page 151 of this Lab Manual.
3. Use yum to attempt to remove the **transmission-common** package to make sure it is not installed. Allow yum to remove any dependencies.

```
$ su -c 'yum remove transmission-common'
```

4. Install the **transmission-gtk** package using the rpm utility; rpm displays a dependency error. See the tip on the preceding page for information on specifying the location of the package.

```
$ su -c 'rpm -ivh /run/media/student/Fedora*/Packages/t/transmission-gtk-*'
error: Failed dependencies:
    transmission-common is needed by transmission-gtk-2.77-3.fc19.i686
```

5. Use rpm to install the package **transmission-gtk** is dependent on. Next, use rpm to install the **transmission-gtk** package.

```
$ su -c 'rpm -ivh /run/media/student/Fedora*/Packages/t/transmission-common-*'
Preparing...                          ################################# [100%]
Updating / installing...
   1:transmission-common-2.77-3.fc19  ################################# [100%]

$ su -c 'rpm -ivh /run/media/student/Fedora*/Packages/t/transmission-gtk-*'
Preparing...                          ################################# [100%]
Updating / installing...
   1:transmission-gtk-2.77-3.fc19     ################################# [100%]
```

6. Use rpm to remove the **transmission-common** package. What happens? Take the steps needed to use rpm to remove the **transmission-common** package.

```
$ su -c 'rpm -e transmission-common'
error: Failed dependencies:
    transmission-common is needed by (installed) transmission-gtk-2.80-2.fc19.i686
```

The rpm utility displays an error message: You cannot remove **transmission-common** without first removing its dependency, **transmission-gtk**.

```
$ su -c 'rpm -e transmission-gtk'
$ su -c 'rpm -e transmission-common'
```

Deliverables

Practice using the rpm utility to install and remove software packages.

LAB 32: INSTALLING SOFTWARE FROM SOURCE CODE FILES (15–30 MINUTES)

OBJECTIVES

In this lab you will learn to install software from source code files.

SETUP

- A Fedora 19 installation that includes a nonprivileged user named **student**. You must know the **root** password.

READING

Read the following section:

- "GNU Configure and Build System" on page 550 of Sobell

PROCEDURE

The yum and rpm utilities install software that has already been compiled. Sometimes an RPM package is not available for a program; other times you may want to compile the software yourself so that you can modify the configuration or add a patch.

1. Download, compile, install, and test the **which** package (which utility).

   a. Before you start, remove the **which** alias so that it is easier to find the new version of which after you install it. Use which to determine which version of which you are using.

   ```
   $ unalias which
   $ which which
   /usr/bin/which
   ```

   b. The make and gcc utilities are required to compile and install software. Install the **make** and **gcc** packages.

   ```
   $ su -c 'yum -y install make gcc'
   ```

   c. Use ftp to log in as an anonymous user and download the source code tar file for the **which** package (**which-2.20.tar.gz**) from the **gnu/which** directory at mirrors.kernel.org.

   ```
   $ ftp mirrors.kernel.org
   Trying 149.20.4.71
   ...
   Connected to mirrors.kernel.org (149.20.4.71).
   220 Welcome to mirrors.kernel.org.
   Name (mirrors.kernel.org:student): ftp
   331 Please specify the password.
   Password: RETURN
   ```

```
...
ftp> cd gnu/which
ftp> get which-2.20.tar.gz
ftp> quit
```

Alternately, you can download the file using a browser.

d. Extract the files from the tar file and use cd to change to the newly created directory. You might want to look at the **README** and **INSTALL** files.

```
$ tar -xvzf which-2.20.tar.gz
$ cd which-2.20
```

e. Configure the source code.

```
$ ./configure
```

f. Compile the source code.

```
$ make
```

g. Install the program.

```
$ su -c 'make install'
```

h. Use which to determine which version of which you are now using.

```
$ which which
/usr/local/bin/which
```

2. Remove the new which program and then display the location of the which utility you will be using.

a. Remove the new which program.

```
$ su -c 'rm /usr/local/bin/which'
```

b. Use which to display the location of the which utility you will be using. If bash reports that it cannot find which, give the command **hash –r** to clear the bash hash table (Sobell, page 376) and try again.

```
$ which which
/usr/bin/which
```

3. How would you determine if the new which program is installed?

You cannot easily query the system for a program installed from source files. The **rpm** setup makes queries a less tedious administration task. In the case of this lab, the best way to determine if the new which program is installed would be to look for the program in the **/usr/local/bin** directory.

DELIVERABLES

Practice downloading, compiling, and installing a program from a source code file.

LAB 33: TROUBLESHOOTING USING rpm QUERIES (20–30 MINUTES)

OBJECTIVES

In this lab you will investigate package contents before and after installation using rpm query and verify commands.

SETUP

- A Fedora 19 installation that includes a nonprivileged user named **student**. You must know the **root** password.
- Access to the Fedora installation DVD or DVD ISO image file

READING

Read the following sections:

- "yum Commands" on page 539 of Sobell
- "RPM: The RPM Package Manager" on page 546 of Sobell
- The yum man page
- The rpm man page

PROCEDURE

1. Mount the Fedora Install DVD or DVD ISO image file as explained in Lab 45 on page 151 of this Lab Manual. Do not set it up as a repository as the subsequent steps of Lab 45 describe. If it is already mounted, continue.
2. Some queries can be performed using either yum or rpm. Compare yum and rpm queries.

   a. List information about the installed **logrotate** package.

   ```
   $ yum list installed logrotate
   $ rpm -q logrotate
   ```

   b. Use wildcards to list information about installed **kernel** packages.

   ```
   $ yum list installed '*kernel*'
   $ rpm -qa | grep kernel
   ```

   c. List information about the **wv** package. List the file on the mounted Fedora Install image file.

   ```
   $ yum list wv
   $ rpm -q wv   #reports package wv is not installed
   $ ls /run/media/student/Fedora*/Packages/w/wv*
   ```

Because two packages in the Install image file begin with **wv**, the last command lists information about two packages, one of which is **wv**.

d. Display information about the **bash** installed package.

```
$ yum info bash
$ rpm -qi bash
```

e. Display information about the **wv** package that is not installed. Some queries are considerably easier using yum.

```
$ yum info wv
$ rpm -qip /run/media/student/Fedora*/Packages/w/wv*rpm
```

f. Find out which package provides the wv utility. Some queries are only possible using yum.

```
$ yum whatprovides bash
$ rpm -q --whatprovides bash
$ rpm -qf /usr/bin/bash   #works only with an installed package
```

g. List the files that are a part of the **logrotate** and **wv** packages. Some queries are only possible using rpm.

```
$ rpm -ql logrotate
$ rpm -qlp /run/media/student/Fedora*/Packages/w/wv*rpm
```

h. Display the pre- and post-install scripts for the **httpd** package.

```
$ rpm -q --scripts httpd
```

i. Display the changelog of the **httpd** package.

```
$ rpm -q --changelog httpd
```

Deliverables

Practice using yum and rpm to display information about software packages.

Lab 34: Learning About GRUB (25–35 minutes)

Objectives

In this lab you will use GRUB to manage the boot process and to pass boot parameters to the kernel.

This Lab Manual refers to GRUB 2 simply as GRUB

tip This Lab Manual refers to GRUB 2 (Grand Unified Boot loader 2) simply as GRUB.

Setup

- A Fedora 19 installation that includes a nonprivileged user named **student**. You must know the **root** password.

Reading

Read the following sections:

- "Modifying Boot Parameters (Options)" on page 70 of Sobell
- "Runlevels" on page 440 of Sobell
- "Booting the System to Single-User/Rescue Mode" on page 450 of Sobell
- "GRUB: The Linux Boot Loader" on page 590 of Sobell

Procedure

1. Display the current and default target unit (runlevel).

   a. Display the current target unit (runlevel; Sobell, page 449).

   ```
   $ who -r
   ```

   or

   ```
   $ runlevel
   ```

   b. Display the persistent (default) target unit (runlevel; Sobell, page 444).

   ```
   $ ls -l /etc/systemd/system/default.target
   ```

2. Edit the GRUB menu to cause the system to boot to the single-user target unit and verify the current target unit.

   a. Edit the GRUB menu so the local system boots to the single-user/rescue target unit (runlevel 1). Follow the instructions starting on page 450 of Sobell. Hold the SPACE bar down, not the SHIFT key, to display the GRUB menu as the system is booting.

   b. When prompted, enter the **root** password.

```
Give root password for maintenance
(or type CONTROL-D to continue): bird482doc
```

c. Display the current target unit; **S** stands for single-user.

```
$ who -r
```

or

```
$ runlevel
```

d. Display the persistent target unit.

```
$ ls -l /etc/systemd/system/default.target
```

e. Reboot the system to verify that the change was temporary. The system should boot to a graphical login prompt.

Type exit or press CONTROL-D at the root prompt while the single-user target unit is active.

f. Log in as **student** and open a terminal emulator window.

3. Install the kernel documentation package (**kernel-doc**).

```
$ su -c 'yum -y install kernel-doc'
```

4. Using the **/usr/share/doc/kernel-doc*/Documentation/kernel-parameters.txt** file, determine a boot parameter to pass to the kernel using GRUB that will

```
$ less /usr/share/doc/kernel-doc*/Documentation/kernel-parameters.txt
```

a. Force the system to boot with SELinux in permissive mode.

```
enforcing=0
```

b. Set the maximum number of processors (CPUs) to use to 4.

```
maxcpus=4
```

c. Prompt for a video mode (VGA) for the text-based terminals.

```
vga=ask
```

d. Load the kernel and start the bash shell instead of the **init** process.

```
init=/usr/bin/bash
```

5. Modify the **/etc/default/grub** file to set the maximum number of processors to 4.

a. Add **maxcpus=4** followed by a SPACE to the beginning of the value assigned to the **GRUB_CMDLINE_LINUX** variable in the **/etc/default/grub** file (Sobell, page 591):

```
GRUB_CMDLINE_LINUX="maxcpus=4 rd.md=0 rd ..."
```

b. After saving a backup copy of **/boot/grub2/grub.cfg**, run grub2-mkconfig (Sobell, page 593) to re-create the file.

```
$ su -c 'cp -a /boot/grub2/grub.cfg /boot/grub2/grub.0'
$ su -c 'grub2-mkconfig -o /boot/grub2/grub.cfg'
```

c. Reboot and display the value of **/proc/cmdline** to verify that the parameter was passed at boot time. See page 512 of Sobell for information on the /**proc** pseudofilesystem.

```
$ cat /proc/cmdline
```

d. Remove the **maxcpus=4** parameter from **/etc/default/grub**, run grub2-mkconfig, and reboot. Check **/proc/cmdline** to make sure the **maxcpus=4** parameter was not passed at boot time.

6. Change the GRUB setup so that GRUB waits forever for the user to select a menu item before booting the system. Test the setup and then change the amount of time GRUB waits before booting the system to 10 seconds and test again.

a. Change the GRUB setup so that GRUB waits forever for user input before booting the system.

Edit the **/etc/default/grub** file and change **GRUB_TIMEOUT=5** to **GRUB_TIMEOUT=–1**.

b. After saving a backup copy of **/boot/grub2/grub.cfg**, run grub2-mkconfig (Sobell, page 593) to re-create the file.

```
$ su -c 'cp -a /boot/grub2/grub.cfg /boot/grub2/grub.1
$ su -c 'grub2-mkconfig -o /boot/grub2/grub.cfg
```

c. Reboot the system and see if GRUB waits forever (give it 15 seconds or so) for you to select an item from the GRUB menu.

d. Highlight the top item on the GRUB menu and press RETURN to boot the system.

e. Change the amount of time GRUB waits before booting the system to 10 seconds, back up **grub.cfg**, run grub2-mkconfig, reboot, and see if GRUB now boots automatically after 10 seconds.

Edit the **/etc/default/grub** file and change **GRUB_TIMEOUT=–1** to **GRUB_TIMEOUT=10**, run grub2-mkconfig, reboot, and see what GRUB does.

Deliverables

Practice working with GRUB, kernel parameters, and a modified GRUB configuration file.

LAB 35: CONTROLLING PROCESSES (30–40 MINUTES)

OBJECTIVES

Monitor, schedule, and send signals to processes.

SETUP

- A Fedora 19 installation that includes a nonprivileged user named **student**. You must know the **root** password.

READING

Read the following sections:

- “tar: Stores or Extracts Files to/from an Archive File” on page 249 of Sobell
- “Process Identification” on page 374 of Sobell
- “kill: Sends a Signal to a Process” on page 465 of Sobell
- “User crontab files” on page 608 of Sobell
- “top: Lists Processes Using the Most Resources” on page 612 of Sobell
- “trap: Catches a Signal” on page 1047 of Sobell

PROCEDURE

1. Launch the GNOME System Monitor by pressing the SUPER (Windows) key and typing **system monitor** followed by a RETURN keystroke. Double-click the titlebar of the System Monitor window to maximize it and make it easier to view.

   a. To display processes, click the tab labeled **Processes**.

   b. Use the View menu at the right end of the toolbar to display All Processes, not just My Processes.

   c. Sort the rows of the report on the column labeled **User** by clicking the column title.

   d. Sort the rows so that the process using the most CPU power (the highest percentage) is at the top. Which process on the system is using the most CPU power?

   Click the **% CPU** column title.

   Varies; probably gnome-system-monitor or gnome-shell.

   e. Which of My Processes is using the most memory?

Use the View menu to display My Processes. Click the **Memory** column title to sort the rows by that column. (You may need to drag the divider that is to the right of the word **Memory** to expand the column so it display units such as MiB.)

Varies; probably gnome-shell.

f. You add a field to the display by right-clicking the Application menu, selecting **Preferences**, clicking the Processes tab, and placing a tick in the check box next to the appropriate line in the Information Fields scrollable list. Add the Status and the (SELinux) Security Context to the display. Click **Close**.

g. Explore other options and tabs in the System Monitor window.

h. Close the System Monitor window.

2. You can use the top utility (Sobell, page 612) in a textual environment to monitor processes. Run top and answer the following questions.

a. Look at the first line that top displays. How long has the local system been running?

Varies.

b. To display the processes being run by a single user, press **u** (user), type the username of the user whose processes you want top to display, and press RETURN. To show all processes again, press **u** immediately followed by a RETURN keystroke.

Which command is the user named **dbus** running?

While running top, press **u**, type **dbus**, and press RETURN. The user named **dbus** is running **dbus-daemon**.

c. Cause top to show all processes.

While running top, press **u** immediately followed by a RETURN.

d. To sort by a column, press **f** (field); top displays the Field Management screen. Use the ARROW keys to move the highlight to the name of the column you want to sort on, press **s** to set the sort field, and press ESCAPE to return to the main top screen.

Which process owned by **student** is using the most memory (**%MEM**)?

While running top, press **u**, type **student**, and press RETURN. Then press **f**, use the ARROW keys to highlight **%MEM**, and press ESCAPE.

Varies, probably gnome-shell.

e. Which process on the system is using the highest percentage of memory?

Continuing from the preceding step, press **u** followed by a RETURN keystroke; top is already sorting by the **%MEM** column.

Varies, probably gnome-shell.

f. Use **h** to display the top Help screen. Answer the following questions based on the information on that screen.

- Which key allows you to terminate (kill; page 465 of Sobell) a process (task)? (You have to own the process to be able to kill it.)

The **k** key allows you to kill a process.

- Which key allows you add the parent process ID (PPID) column to the information that top displays?

The **f** or **F** key allows you to add a column.

- Which key allows you save the current configuration of top so that it displays the same information next time you run it?

The **W** key writes (saves) the current configuration information.

- Which key exits from top?

The **q** key exits from top.

g. Exit from top.

3. You can list process information from the command line using the ps utility (Sobell, page 374).

a. By default, ps displays the processes started from the shell it is run from. What is the process ID (PID) number of the shell you are running?

Varies. A ps command displays the PID number of the shell it is run from in the column labeled **PID** in the row that has the value **bash** in the column labeled **CMD**.

b. Display information about all processes by giving the command **ps –ef** or **ps –aux**.

c. To display only processes owned by one user, give ps an argument of **–U** immediately followed by the username of that user. Display processes owned by **student**.

```
$ ps -Ustudent
```

d. To display all processes running a specific utility, send the output of **ps –ef** through a pipeline to grep with an argument of the name of the utility you are searching for. Display all processes running bash.

```
$ ps -ef | grep bash
```

e. You can use the ps **–o** option to customize the output of ps. To list all processes displaying only the UID (user ID) number, PID (process ID) number, and command being run by the process, specify the following strings, separated by commas, following the **–o** on the command line: **uid**, **pid**, **comm**.

Display the UID, PID, and command for all processes that were started from the current shell.

```
$ ps -ouid,pid,comm
```

f. For all processes running on the system, list the PID number, PPID (parent process ID; **ppid**) number, username (**user**) of the owner, and command being run by the process.

```
$ ps -e -opid,ppid,user,comm
```

4. Frequently, you can terminate a hung process by sending it a signal using the kill utility (Sobell, page 465).

a. Run the sleep utility in the background by giving the following command

```
$ sleep 100 &
```

b. Use ps to display the PID number of the process running sleep. Terminate the process running sleep by using kill with the default TERM signal.

```
$ ps
$ kill PID    # PID is the value that ps displays
```

c. Open a second terminal emulator window and kill the shell running in that window from the original terminal emulator window.

- Open a second terminal emulator window by pressing CONTROL-SHIFT-N from the first terminal emulator window. Display the PID number of the bash process running in the *new* terminal window.

Give the following command from the second terminal emulator window.

```
$ ps
```

- Click in the first terminal emulator window to make it the active window. Give a kill command to terminate the bash process running in the second terminal emulator window. What happens? Which signal will force the bash process to terminate (and close the window)?

Nothing happens when you do not specify a signal (kill uses the TERM [signal 15] signal by default). You must use the KILL (signal 9) signal to kill the bash process running in the second terminal emulator window.

```
$ kill -KILL PID
```

5. Modify the **report.sh** script (Lab Manual, page 102) so that it also lists all the bash processes running on the system. Set up the script so that it displays the username of the owner, PID number, terminal (**tty**), and command.

Add the following line to the script

```
ps -e -ouser,pid,tty,comm | grep bash
```

6. Working as **student**, use crontab to schedule **report.sh** to run every Mon/Wed/Fri at 8:00 AM (Sobell, "User crontab files," page 608).

   Edit the **crontab** file by giving the command **crontab –e** and add the following line to the file:

   ```
   0 8 * * 1,3,5 /home/student/bin/report.sh
   ```

7. Working as **student**, set up an automated backup of your home directory.

   a. Use tar (Sobell, page 249) to write the contents of your home directory to a file named **/tmp/student_home.tar.**

   ```
   $ tar -cvf /tmp/max_home.tar ~
   ```

   b. Modify the tar command to add gzip compression (Sobell, page 252) and name the file **/tmp/student_home.tgz**.

   ```
   $ tar -czvf /tmp/max_home.tgz ~
   ```

   c. Modify the command to use command substitution (Sobell, page 410) to add the current date to the name of the output file. (*Hint:* The command **date +%Y%m%d** outputs an appropriately formatted date.)

   ```
   $ tar -czvf /tmp/student_home-$(date +%Y%m%d).tgz ~
   ```

   d. Schedule the final tar command to run every Saturday at 10:00 PM.

   Edit the **crontab** file by giving the command **crontab –e** and add the following line to the file:

   ```
   0 10 * * 6 tar czvf /tmp/student_home-$(date +%Y%m%d).tgz ~
   ```

DELIVERABLES

Experience running the GNOME System Monitor, top, and ps, as well as a modified report script that runs regularly and includes process monitoring.

Lab 36: Troubleshooting Network Connections (5–10 minutes)

Objectives

In this lab you will use CLI tools to learn about network connections.

Setup

- A Fedora 19 installation that includes a nonprivileged user named **student**. You must know the **root** password.
- Access to the classroom server. See the tip on page 73 of this Lab Manual.

Reading

Read the following sections:

- “ping: Tests a Network Connection” on page 305 of Sobell
- “Subnet mask” on page 298 of Sobell
- “traceroute: Traces a Route over the Internet” on page 306 of Sobell
- “host and dig: Query Internet Nameservers” on page 307 of Sobell
- dig utility on pages 861, 862, 866, and 885 of Sobell

Procedure

1. Use ping (Sobell, page 305) to test the connections to several systems.
   a. See step 5d of Lab 3 on page 16 of this Lab Manual to determine the IP address of the local system and that of the DNS server. Write down these addresses.
   b. How many milliseconds (ms), on average, does it take for the local system to return a ping?

      Give the following command, substituting the IP address of the local system for *xxx.xxx.xxx.xxx*. The last line of the output shows the average round trip time (rtt) for each ping.

      ```
      $ ping -c 4 xxx.xxx.xxx.xxx
      ```

   c. Can you ping the classroom server?

      ```
      $ ping class
      ```

   d. Some sites do not respond to ping. Try to ping www.microsoft.com.
   e. Can you ping the DNS server?

f. Can you ping www.sobell.com?

2. Compare the output of the dig and host domain name utilities.

a. Use host to display information about www.fedoraproject.org.

```
$ host www.fedoraproject.org
```

b. Use dig to display information about www.fedoraproject.org.

```
$ dig www.fedoraproject.org
```

c. Try using **host** to display information about the nonexistent domain named **srv**. This test will fail because the domain does not exist.

```
$ host srv
Host srv not found: 3(NXDOMAIN)
```

3. Display the IP address and subnet mask (Sobell, page 298) of the local system using the ip utility.

The following command displays information about two network connections. The first is localhost (**lo**, the loopback service). The second is the external connection this step asks about. The value following **inet** is the IP address of the local system. This value is followed by a slash and the subnet mask.

```
$ ip addr
```

4. *Optional:* Use traceroute (Sobell, page 306) to display the path a packet takes from the local system to www.fedoraproject.org.

```
$ traceroute www.fedoraproject.org
```

Deliverables

Practice using the ping, dig, host, and traceroute utilities to display information about network connections.

LAB 37: CONFIGURING AN FTP SERVER (10–20 MINUTES)

OBJECTIVES

In this lab you will set up an FTP server.

SETUP

- A Fedora 19 installation that includes a nonprivileged user named **student**. You must know the **root** password.

READING

Read the following section:

- "Setting Up an FTP Server (vsftpd)" on page 724 of Sobell

PROCEDURE

1. Install, configure, and test the **vsftpd** FTP server.

   a. Make sure the **vsftpd** package is installed. If **vsftpd** is not installed, install it.

   ```
   $ rpm -q vsftpd
   $ su -c 'yum -y install vsftpd'
   ```

   b. Working with **root** privileges, set up the **vsftpd** service to start each time the system enters multiuser mode and then start the **vsftpd** service.

   ```
   $ su -c 'systemctl enable vsftpd.service'
   $ su -c 'systemctl start vsftpd.service'
   ```

   c. Add content to the **/var/ftp/pub** directory by copying the **/user/share/dict/linux.words** file to the **/var/ftp/pub** directory.

   ```
   $ su -c 'cp /usr/share/dict/linux.words /var/ftp/pub'
   ```

   d. Add more content by copying the **/usr/share/doc/vsftpd*** directory hierarchy to the **/var/ftp/pub** directory. Use the cp **–r** option to copy the directory and its contents.

   ```
   $ su -c 'cp -r /usr/share/doc/vsftpd* /var/ftp/pub'
   ```

   e. Test the FTP server by using ftp to connect to the local system, first as **student**, then as an anonymous user (**anonymous** or **ftp**). See "JumpStart II: Starting a **vsftpd** FTP Server" and "Troubleshooting" both on page 725 of Sobell.

   Give the following command and log in as **student**, supplying the password for **student**. Give a **quit** command to exit from ftp. Then give the following command again, this time logging in as **anonymous** or **ftp**; any password (or no password) will work. Exit from ftp.

```
$ ftp localhost
```

2. Set up the server (local system) so that it can accept connections from remote systems and so that it allows local users to log in.

   a. Open the FTP port using firewalld-cmd so that remote systems can access the FTP server.

   ```
   $ su -c 'firewall-cmd --add-service=ftp'
   $ su -c 'firewall-cmd --permanent --add-service=ftp'
   ```

   b. Give the following command to cause SELinux to allow users to access their home directories using FTP.

   ```
   $ su -c 'setsebool -P ftp_home_dir on'
   ```

   Alternately, you can run SELinux in permissive mode by giving the following command.

   ```
   $ su -c 'setenforce permissive'
   ```

   Each of these commands must be run with **root** privileges.

   c. The Fedora **vsftpd** configuration file sets **local_enable** to **YES** (Sobell, page 728), which allows local users to log in on the FTP server. If you were to change the **vsftpd** file, you would need to restart the **vsftpd** daemon. Because of the way Fedora sets up **vsftpd**, you do not need to change the configuration file and you do not need to restart the server.

Deliverables

An FTP server offering public content for local and anonymous connections. In the next lab you will work with the FTP server from a remote system.

LAB 38: EXPLORE FTP CLIENT UTILITIES (10–20 MINUTES)

OBJECTIVES

In this lab you will learn to use the ftp and lftp FTP client utilities.

SETUP

- A Fedora 19 installation that includes a nonprivileged user named **student**. You must know the **root** password.
- Access to the classroom server. See the tip on page 73 of this Lab Manual.

READING

Read the following sections:

- "Introduction to FTP" on page 714 of Sobell
- "Running the ftp and sftp FTP Clients" on page 716 of Sobell
- "Tutorial Session" on page 717 of Sobell.

PROCEDURE

1. Practice copying files to and from a remote system using FTP.

   a. Use vi to create a short file named **memo.local** in your home directory on the local system. You will use FTP to copy this file to the remote system.

   ```
   $ vi ~/memo.local
   ```

   b. Give an **ip addr** command and make note of the IP address of the local student) system. Use this address in place of *myIP* throughout this lab.

   c. Use ssh to connect to the classroom server as the user **rose** with a password of **fit714tree**.

   ```
   $ ssh rose@class
   ```

   d. Use vi to create a short file named **memo.class.***XXX*, where *XXX* is your initials, in Rose's home directory on the classroom server.

   ```
   [rose@classroom ~]$ vi ~/memo.class.XXX
   ```

   e. Working from the classroom server, connect to the FTP server running on the local (student) system and authenticate as **student**.

   ```
   [rose@classroom ~]$ ftp myIP
   Connected to myIP (myIP).
   220 (vsFTPd 3.0.2)
   Name (myIP:rose): student
   331 Please specify the password.
   ```

```
Password:fit714tree
230 Login successful.
Remote system type is UNIX.
Using binary mode to transfer files.
ftp>
```

f. List the files in your home directory (**~student**) on the local (student) system. Remember, ftp is now connected to the local (student) system (the FTP server).

```
ftp> ls
```

g. Use the ftp **get** command to copy the **memo.local** file from your home directory on the local (student) system (the FTP server) to Rose's home directory on the classroom server. The ftp client is running on the classroom server and is connected to the local system (the FTP server) so you are getting the file from the "remote system" (the local system) and copying it to the classroom server.

```
ftp> get memo.local
```

h. Use the ftp **put** command to copy the **memo.class.*XXX*** file from Rose's home directory on the classroom server to your home directory on the local system.

```
ftp> put memo.class.XXX
```

i. Exit from ftp but remained logged in on the classroom server as Rose.

```
ftp> quit
```

2. Working on the classroom server, use ftp to copy the **/etc/hosts** file from the classroom server to Ben's home directory on the local system.

a. While logged in on the classroom server, connect to the FTP server running on the local (student) system and authenticate as **ben**.

```
[rose@classroom ~]$ ftp myIP
Connected to myIP (myIP).
220 (vsFTPd 3.0.2)
Name (myIP:rose): ben
331 Please specify the password.
Password:fit714tree
230 Login successful.
Remote system type is UNIX.
Using binary mode to transfer files.
ftp>
```

b. Use the ftp **put** command to copy the **/etc/hosts** file from the classroom server to Ben's home directory on the local system. Which error does ftp display? Why do you get this error?

```
> ftp put /etc/hosts
local: /etc/hosts remote: /etc/hosts
```

```
227 Entering Passive Mode (192,168,206,136,107,237).
553 Could not create file.
```

The ftp utility displays a **Could not create file error** because it (running as Ben) does not have permission to overwrite the existing **/etc/hosts** file on the local system. Note the use of an absolute pathname.

c. Try again, specifying the simple filename, **hosts.class**, as the destination file.

```
ftp> put /etc/hosts hosts.class
```

d. Another way to copy the same file is to change to the **/etc** directory before you copy the **hosts** file. That way you can specify a simple filename and do not need to specify a destination file.

```
ftp> lcd /etc
ftp> put hosts
```

e. Exit from ftp but remained logged in on the classroom server as Rose.

```
ftp> quit
```

3. Explore an anonymous connection.

a. Once again, use ftp to connect to the local system; log in as the user named **anonymous** (you can also specify the username **ftp**). Traditionally the password is your email address. Pressing RETURN without entering a password also works.

```
[rose@classroom ~]$ ftp myIP
```

b. Issue the command **ls /**. Which files are in the root directory?

Because **vsftpd** sets up a chroot jail (Sobell, page 487) for an anonymous user, all that appears in the root directory is the **pub** directory. On the server, this root directory equates to the **/var/ftp** directory.

c. Which files are in the **pub** directory?

```
ftp> ls pub
```

d. Change to the **pub/vsftpd*** directory. You must specify the exact name of the directory, you cannot use a wildcard character (you cannot use *).

```
ftp> cd pub/vsftpd-3.0.2
```

The version number on the local system may be different from that in the example. The exact name of the **vsftpd*** directory is listed in step c.

e. While still working in ftp, create a **~/vsftpd.*XXX*** directory in Rose's home directory on the classroom server, where *XXX* are your initials. Use the ftp **lcd** command to change directories to the directory you just created.

```
ftp> !mkdir ~/vsftpd.STG
ftp> lcd ~/vsftpd.STG
Local directory now /home/rose/vsftpd.STG
```

f. Use the ftp **mget** command to transfer all the individual files from the **vsftpd*** directory on the local (student) system (the FTP server) to this new **vsftpd** directory. When ftp prompts you (asks if you want to copy a file), press CONTROL-C; when ftp asks if you want to continue using **mget**, answer **no**.

```
ftp> mget *
mget AUDIT? CONTROL-C
Continue with mget? n
ftp>
```

g. Turn off prompting and copy the files.

```
ftp> prompt
Interactive mode off.
ftp> mget *
```

h. Exit from ftp but remained logged in on the classroom server as Rose.

4. Use the lftp utility to connect to the local system and give a few commands.

a. Give an lftp command to connect to the local system.

```
$ lftp myIP
lftp myIP:~>
```

b. Did you have to supply a username and password?

No. When you do not specify a username on the command line, lftp logs you in as an anonymous user automatically.

c. List the contents of the **pub/vsftpd*** directory. (You can use command completion with lftp: type enough of the pathname to uniquely identify the directory and then press TAB.)

```
lftp myIP:~> ls pub/vsTAB
```

d. Press the UP ARROW and DOWN ARROW keys to scroll through the previous commands.

e. Exit from ftp and then log off of the classroom server so you are working as **student** on the local system.

```
lftp myIP:~>quit
[rose@classroom ~]$ exit
logout
Connection to class closed.
[student@linux ~]$
```

Deliverables

Practice using ftp and lftp to list and copy files to and from a remote system.

Lab 39: Connecting to NFS Shares (20–30 minutes)

Objectives

In this lab you will mount remote NFS shares on the local system.

Setup

- A Fedora 19 installation that includes a nonprivileged user named **student**. You must know the **root** password.
- Access to the classroom server. See the tip on page 73 of this Lab Manual.

Reading

Read the following sections:

- "Introduction to NFS" on page 803 of Sobell
- "Running an NFS Client" on page 805 of Sobell

Procedure

1. Make the **/var/ftp/pub** directory on the classroom server available on the local system as **/nfsmount**.

   a. Create the **/nfsmount** directory.

   ```
   $ su -c 'mkdir /nfsmount'
   ```

   b. Mount the **/var/ftp/pub** directory from the classroom server on the **/nfsmount** mount point on the local (student) system.

   ```
   $ su -c 'mount class:/var/ftp/pub /nfsmount'
   ```

2. Explore access to the files in the **/nfsmount** directory hierarchy.

   a. Who owns the files in **/nfsmount**?

   ```
   $ ls -Rl /nfsmount | less
   ```

   The files in **/nfsmount** are owned by **root**.

   b. Can **student** read the files? Modify the files? Create or delete a file?

   ```
   $ tail /nfsmount/themes/Emacs/gtk-3.0/gtk-keys.css
   $ echo hi >> /nfsmount/themes/Emacs/gtk-3.0/gtk-keys.css
   $ rm /nfsmount/themes/Emacs/gtk-3.0/gtk-keys.css
   $ touch /nfsmount/somefile
   ```

   The user named **student** can read the files in **/nfsmount**, but cannot modify, create, or delete files in this directory.

c. Can a user working with **root** privileges read the files? Modify the files? Create or delete a file?

A user working with **root** privileges can read the files in **/nfsmount**. By default, NFS mounts directories with the **root_squash** option (Sobell, page 817), so a user working with **root** privileges works as the user named **nfsnobody** and cannot modify, create, or delete files in **/nfsmount**.

3. Make the files available each time the local system boots.

a. Edit the **/etc/fstab** file and add the following line to the file

```
class:/var/ftp/pub /nfsmount nfs defaults 0 0
```

b. Reboot the local system and make sure the files under **/nfsmount** are available.

```
$ ls -R /nfsmount
```

4. Mount the **/testing** directory hierarchy from the classroom server on the **/nfsrw** mount point on the local (student) system; the files in this directory are readable and writable. Explore access to the files in **/nfsrw**.

a. Create **/nfsrw** and mount **/testing** from the classroom server on it.

```
$ su -c 'mkdir /nfsrw'
$ su -c 'mount class:/testing /nfsrw'
```

b. Who owns the files in **/nfsrw**? What are the permissions on these files?

```
$ ls -l /nfsrw
```

The files in **/nfsrw** are owned by **student**. They are readable and writable by owner, group, and readable by other (664 permissions). That is, everyone can read from these files and student and other can write to them.

c. Working as **student**, try to copy **/nfsrw/memo1** to **/nfsrw/memo1.*XXX***, where ***XXX*** are your initials. What happens? Who owns the copied file?

```
$ cp /nfsrw/memo1 /nfsrw/memo1.STG
$ ls -l /nfsrw/memo1.STG
```

There is no problem copying the file. The user who copied the file, **student**, owns the copied file. Replace **STG** in the example with your initials.

d. Working with **root** privileges, copy **/nfsrw/memo2** to **/nfsrw/memo2.*XXX***, where ***XXX*** are your initials. Who owns the copied file? Why?

```
$ su -c 'cp /nfsrw/memo2 /nfsrw/memo2.STG'
$ ls -l /nfsrw/memo2.STG
```

The user named **nfsnobody** owns the file because by default, NFS mounts directories with the **root_squash** option (Sobell, page 817).

e. Unmount the **/nfsrw** directory

```
$ su -c 'umount /nfsrw'
```

Deliverables

Practice mounting remote NFS files and a remote NFS directory mounted on the local system.

Lab 40: Sharing Files Using NFS (20–30 minutes)

Objectives

You will create read and read/write NFS shares.

Setup

- A Fedora 19 installation that includes a nonprivileged user named **student**. You must know the **root** password.
- Access to the classroom server. See the tip on page 73 of this Lab Manual.

Reading

Read the following section:

- “Setting Up an NFS Server” on page 811 of Sobell

Procedure

1. Set up NFS.

   a. Use rpm to make sure that the **nfs-utils** package is installed.

   ```
   $ rpm -q nfs-utils
   ```

   b. Make sure **rpcbind** is running.

   ```
   $ systemctl status rpcbind.service
   ```

   c. Set up the nfs-server.service service to start each time the system enters multiuser mode.

   ```
   $ su -c 'systemctl enable nfs-server.service'
   ```

2. Export a directory, start the NFS server, and set up the firewall.

   a. Working with **root** privileges, use an editor to modify the **/etc/exports** file to share the **/usr/share/backgrounds** directory with the classroom server with readonly access. Because of a bug, you cannot specify the classroom server as **class**, but must specify its IP address, which you can obtain from the **/etc/hosts** file.

   ```
   $ grep class /etc/hosts
   $ su -c 'vi /etc/exports'
   $ cat /etc/exports
   /usr/share/backgrounds classroom-server-IP(ro,sync)
   ```

   b. Use exportfs to export the directory and to display information about the exported directory.

```
$ su -c 'exportfs -ra'
$ su -c exportfs
```

c. Use firewall-cmd to open ports TCP 2049 and UDP 2049 in both the running and permanent configurations. See “Opening a Port” on page 908 of Sobell for more information.

```
$ su -c 'firewall-cmd --add-port=2049/tcp'
$ su -c 'firewall-cmd --permanent --add-port=2049/tcp'
$ su -c 'firewall-cmd --add-port=2049/udp'
$ su -c 'firewall-cmd --permanent --add-port=2049/udp'
```

d. Start the nfs-server.service service.

```
$ su -c 'systemctl start nfs-server.service'
```

Special setup for this lab

tip The **sudoers** file (Sobell, page 433) on the classroom server has been set up to allow the user named **rose** to run mkdir with **root** privileges and to run mount and umount to work with directories with the name **/bkgnds***.

3. Working on the classroom server, mount the exported directory and copy a file from that directory to Rose’s home directory on the classroom server.

a. Give an **ip addr** command and make note of the IP address of the local (student) system. Use this address in place of *myIP* throughout this lab.

b. Use ssh to log in on the classroom server. Authenticate as Rose.

```
$ ssh rose@class
```

c. Working as Rose on the classroom server, create a directory on which to mount the directory that you exported from the local system. Using sudo to work with **root** privileges, create the **/bkgnds.*XXX*** directory, where *XXX* are your initials.

```
[rose@class ~]$ sudo mkdir /bkgnds.XXX
```

d. Using sudo to work with **root** privileges, mount the directory hierarchy that you exported on the local (student) system. Mount it on **/bkgnds.*XXX***.

```
[rose@class ~]$ sudo mount myIP:/usr/share/backgrounds /bkgnds.XXX
```

e. Display the newly mounted directory using df and ls.

```
[rose@class ~]$ df -h /bkgnds.XXX
[rose@class ~]$ ls /bkgnds.XXX
```

f. Still working on the classroom server, copy **/bkgnds.*XXX*/gnome/Blinds.jpg** to Rose’s home directory. As you copy it, give it the name **blinds.*XXX*.jpg** where *XXX* are your initials.

```
[rose@class ~]$ cp /bkgnds.XXX/gnome/Blinds.jpg ~/blinds.STG.jpg
```

Substitute your initials for **STG** in the preceding command.

g. Using sudo to work with **root** privileges, unmount the directory hierarchy that you mounted on on **/bkgnds.*XXX***.

```
[rose@class ~]$ sudo umount /bkgnds.XXX
```

h. Exit from the classroom server and return to working as **student** on the local system.

```
[rose@class ~]$ exit
```

4. *Optional:* Explore the information provided by **exportfs –v** (Sobell, page 818) and **rpcinfo –p** (Sobell, page 820).
5. *Optional:* Install and explore the capabilities of the **system-config-nfs** utility (Sobell, page 812).

DELIVERABLES

Experience setting up an NFS server and accessing it from a remote client.

LAB 41: EXPLORING ON-DEMAND MOUNTING
(20–30 MINUTES)

OBJECTIVES

In this lab you will configure the local system to act as a client to a remote NFS share that is mounted on demand.

SETUP

- A Fedora 19 installation that includes a nonprivileged user named **student**. You must know the **root** password.
- Access to the classroom server. See the tip on page 73 of this Lab Manual.

READING

Read the following section:

- “automount: Mounts Directory Hierarchies on Demand” on page 821 of Sobell

PROCEDURE

1. Configure the local system to use **autofs** to monitor the **/themes** directory and mount the **/var/ftp/pub/themes** directory from the classroom server on **/themes** on demand.

   a. Install, enable, and start the **autofs** service.

   ```
   $ su -c 'yum -y install autofs'
   $ su -c 'systemctl enable autofs.service'
   $ su -c 'systemctl start autofs.service'
   ```

   b. Create the **/themes** directory.

   ```
   $ su -c 'mkdir /themes'
   ```

   c. Edit the **/etc/auto.master** file and add the following line:

   ```
   /-          /etc/auto.misc --timeout=30
   ```

   d. Edit the **/etc/auto.misc** file and add the following line:

   ```
   /themes    -fstype=nfs4 class:/var/ftp/pub
   ```

   e. Use systemctl to reload the **autofs** service so that it rereads the modified configuration files.

   ```
   $ su -c 'systemctl reload autofs.service'
   ```

2. Test the setup.

a. Use mount and grep to see if any files from the classroom server (**class**) are mounted. You cannot simply list **/themes** as the next step does, because that would mount the remote directory.

```
$ mount | grep class
```

b. To mount the directory, use ls to list the contents of the **/themes** directory. Use mount again to see if any files from class are mounted.

```
$ ls -R /themes
$ mount | grep class
```

c. Wait the amount of time specified by the **––timeout** option in the **/etc/auto.master** file.

```
$ sleep 30
```

d. Is the **/themes** directory still mounted?

```
$ mount | grep class
```

DELIVERABLES

A system mounting a remote NFS share on demand.

Lab 42: Sharing Local Directories Using Samba (20–30 minutes)

Objectives

You will configure a Samba server to use for file sharing.

Setup

- A Fedora 19 installation that includes a nonprivileged user named **student**. You must know the **root** password.

Reading

Read the following sections:

- "Introduction to Samba" on page 828 of Sobell
- "Working with Shares from Linux" on page 832 of Sobell
- "Prerequisites" on page 836 of Sobell
- "**smb.conf**: Manually Configuring a Samba Server" on page 839 of Sobell up to "Parameters in the **smbd.conf** File" on page 841
- "Troubleshooting" on page 846 of Sobell

Procedure

1. Set up a Samba server on the local system.

   a. Install the **samba** package.

   ```
   $ su -c 'yum -y install samba'
   ```

   b. Enable and start the **smb** and **nmb** services.

   ```
   $ su -c 'systemctl enable smb.service'
   $ su -c 'systemctl enable nmb.service'
   $ su -c 'systemctl start smb.service'
   $ su -c 'systemctl start nmb.service'
   ```

   c. Set up SELinux so that it allows Samba clients to browse home directories. This command will take a while to complete.

   ```
   $ su -c 'setsebool -P samba_enable_home_dirs on'
   ```

   d. Add a Samba password of **fit714tree** for **student**.

   ```
   $ su -c 'smbpasswd -a student'
   ```

2. Use smbclient to browse the shares on the local system.

a. Use the smbclient **–L** (list) option to list the shares on the local system (**localhost**) as an anonymous user (do not enter a password, just press RETURN).

```
$ smbclient -L localhost
```

b. Use the smbclient **–U** (user) option to browse the shares on the local system (**localhost**) as **student** (enter the password for **student** you created using smbpasswd). List the files in the home directory of **student** and then exit from smbclient.

```
$ smbclient //localhost/student -U student
Enter student's password: fit714tree
Domain=[MYGROUP] OS=[Unix] Server=[Samba 4.0.13]
smb: \> ls
...
exit
```

The smb.conf file includes a lot of commented-out examples

tip The **/etc/samba/smb.conf** sample configuration file contains extensive comments and commented-out examples. Comment lines start with either a hashmark (**#**) or a semicolon (**;**). The sample file uses hashmarks to begin lines that are intended to remain as comments. Semicolons begin lines you might want to mimic or use as is by removing the semicolons.

3. Add a new share, test the configuration file syntax, and browse the new share.

a. Add a new share to the end of the **/etc/samba/smb.conf** file that allows public readonly access to the **/tmp** directory. Name the share **TEMP** and include **public share of /tmp** as a comment. See “Basic **smb.conf** file” on page 840 of Sobell.

```
[TEMP]
   comment = public share of /tmp
   public = yes
   writable = no
   path = /tmp
```

Your answer might differ. For example, **writable = no** is the same as **read only = yes**.

b. Use testparm to check the configuration file syntax.

```
$ testparm
```

c. Browse again as the anonymous user and see if smbclient lists the **TEMP** share. You do not need to reload the configuration files after you change **smb.conf**.

```
$ smbclient -L localhost
```

To access Samba shares from a remote system, open the firewall

tip If you want to access the Samba shares from a remote system, you must open the firewall. See "Server firewall" on page 836 of Sobell.

4. Add another share.

   a. Give the share the following characteristics.

   - Name the share **Past**.
   - Add the comment **Files for the Pasture staff**.
   - Share the **/tmp/shared** directory.
   - Allow anyone to read the share.
   - Display the share in the browse list.
   - Allow the members of the group named **staff** to write to the share.

```
[Past]
   comment = Files for the Pasture staff
   path = /tmp/shared
   public = yes
   browseable = yes
   write list = +staff
```

   b. Use smbclient to browse the **Past** share.

```
$ smbclient -L localhost
```

Deliverables

A system acting as a Samba server for a readonly share and a share with selective write access.

LAB 43: CONNECTING TO EXISTING SAMBA SHARES (15–25 MINUTES)

OBJECTIVES

You will configure a client to connect to CIFS (Samba) shares.

SETUP

- A Fedora 19 installation that includes a nonprivileged user named **student**. You must know the **root** password.
- Access to the classroom server. See the tip on page 73 of this Lab Manual.

READING

Read the following section:

- "Working with Shares from Linux" on page 832 of Sobell

PROCEDURE

1. Make sure that the **samba-client** package is installed. Install it if it is not installed.

```
$ rpm -q samba-client
```

2. Use smbclient to browse the shares on the classroom server.

   a. Use the smbclient **–L** (list) option to list the shares on the classroom server (**class**) as an anonymous user (do not enter a password, just press RETURN). What is the name of the server? What is the name of the workgroup?

   ```
   $ smbclient -L class
   ```

   The name of the server is **CLASS** and the workgroup is **MYGROUP**.

   b. Use the smbclient **–L** (list) and **–U** (user) options to list the shares on the classroom server (**class**) as **rose** (password **fit714tree**). Which additional shares does smbclient display?

   ```
   $ smbclient -L class -U rose
   ```

   The smbclient utility displays the shares named **rose** and **fake** (the share for the printer).

   c. Use the smbclient **–U** (user) option to browse the shares on the classroom server (**class**) as **rose**. List the files in the home directory of **rose** and exit from smbclient.

```
$ smbclient //class/rose -U rose
Enter rose's password: fit714tree
Domain=[MYGROUP] OS=[Unix] Server=[Samba 4.0.13]
smb: \> ls
...
exit
```

d. *Optional:* The smbclient utility uses many of the same commands as an FTP client. Use **ls**, **cd**, **get**, and **put** to manage files. You can give the command **!ls** in response to the **smb: \>** prompt to list the files in the working directory on the local system. Exit from smbclient when you are finished.

3. Connect to the share named **rose** on the classroom server.

a. Open a Nautilus File Browser window on the local system by giving the command **nautilus** on the command line.

b. Follow the instructions in the section under "Browsing Shares" on page 833 of Sobell. Specify a server address of **smb://class/rose** and, when prompted, a username of **rose** and a password of **fit714tree**. The domain (workgroup) is **MYGROUP**.

The browser displays the contents of Rose's home directory.

Close the Nautilus File Browser window.

c. The **mount.cifs** command is equivalent to the **mount -t cifs** command; both mount a Samba share on the local system. Mount the **rose** share from the classroom server on **/roseshare** on the local system and list the contents of **/roseshare**. When you are finished, unmount the **/roseshare** directory.

```
$ su -c 'mkdir /roseshare'
$ su -c 'mount.cifs -o user=rose //class/rose /roseshare'
$ ls /roseshare
$ su -c 'umount /roseshare'
```

Deliverables

A system acting as a client to a remote Samba share.

Lab 44: Installing and Configuring a DNS Cache (a Caching-Only Nameserver) (30–40 minutes)

Objectives

In this lab you will configure a DNS cache (a caching-only nameserver).

Setup

- A Fedora 19 installation that includes a nonprivileged user named **student**. You must know the **root** password.

Reading

Read the following sections:

- "Introduction to DNS" on page 852 of Sobell
- "Prerequisites" on page 864 of Sobell
- "Notes" on page 865 of Sobell
- "JumpStart I: Setting Up a DNS Cache" on page 866 of Sobell

Procedure

This lab does not set up a static IP address required by a real server

tip A server needs a static (constant) IP address so remote systems can locate it on the Internet or LAN. The local (student) system uses DHCP to acquire an IP address. DHCP provides a dynamic address (an address that might not stay the same over time). Step 1b of this lab makes a change to the **/etc/resolv.conf** file, a change that will not survive a reboot. Normally you would not edit this file because it is overwritten by NetworkManager each time the system boots. This lab edits this file so you can see how a DNS cache works without setting up a static IP address. See page 641 of Sobell for instructions on configuring a static IP address.

1. Set up a DNS cache.

   a. Install the **bind** package.

   ```
   $ su -c 'yum -y install bind'
   ```

   b. To make this DNS cache work without setting up a static IP address, you must add a **nameserver** line to the **/etc/resolv.conf** file. Adding this line is a temporary fix that allows you to experiment with the DNS cache. As soon as you reboot the system, NetworkManager removes the line you added from **resolv.conf**. See the preceding tip.

Add the following line to the **/etc/resolv.conf file**. The new line must be the first **nameserver** line in the file (it must precede any other **nameserver** lines).

```
nameserver 127.0.0.1
```

c. Enable and start the **named** service.

```
$ su -c 'systemctl start named.service'
$ su -c 'systemctl enable named.service'
```

d. Use dig to look up information about fedoraproject.org. How long did the query take (Query time)?

```
$ dig fedoraproject.org
```

The initial query generally takes hundreds of milliseconds (several tenths of a second).

e. Use dig to look up information about fedoraproject.org again. How long did the query take this time? What caused the difference in query times?

```
$ dig fedoraproject.org
```

Subsequent queries generally take a few milliseconds. The first query finds the IP address from an authoritative DNS server on the Internet; it requires **named** to connect to the Internet and query a server. That query stores the result (the IP address) in the local DNS cache. Subsequent queries can quickly access the local DNS cache and so do not take as long to complete.

Deliverables

A temporary caching-only nameserver.

LAB 45: *OPTIONAL:* SETTING UP THE INSTALL MEDIA AS A REPOSITORY (10–15 MINUTES)

OBJECTIVES

To update or install software on the local system using yum, a collection of software packages called a *repository* must be made available to the local system and the local system must know where to find the repository. During installation, Fedora 19 configures the system to look for a repository on the Internet at any Fedora mirror site. Many companies have systems behind firewalls and do not allow Internet access. This lab sets up the Fedora Install DVD or ISO image file as a repository.

1. Make the Fedora Install DVD or ISO image file available as a repository.

   a. **Follow these instructions if the student system is running on a VM:**

   Display the Virtual Machine Settings window by clicking **VMware Player menubar: Virtual Machine⇨Virtual Machine Settings** and then click the CD/DVD (IDE) tab on the left side of the window. Make sure the boxes labeled **Connected** and **Connect at power on** have ticks in them.

   - If you installed Fedora from the Install DVD, insert the Install DVD in the DVD drive on the local system, click the radio button labeled **Use a physical drive**, and make sure **Auto detect** is displayed in the drop-down list labeled **Device**.
   - If you installed Fedora from an ISO image file, click the radio button labeled **Use ISO image** and make sure the file you installed Fedora from is displayed in the drop-down list immediately below that radio button.

   Click **Save**.

   b. **Follow these instructions if the student system is running on a stand-alone physical system:**

   - Insert the Fedora Install DVD in the DVD drive on the local system. Fedora will ask if you want to mount the DVD; answer yes.

2. Open a terminal emulator window (Lab Manual, page 15).
3. Determine where the DVD or ISO image file is mounted by giving the following command.

```
$ df | grep Fedora
/dev/sr0                4412948 4412948         0 100% /run/media/student/Fedora 19 i386
```

The df utility will display the mount point; the preceding command shows the disk/image mounted at **/run/media/student/Fedora 19 i386**. Because this pathname contains SPACEs, you must quote it when you type it on the command line.

a. Working with **root** privileges, disable the Internet repository mirrors by moving the repository configuration files to **root**'s home directory (**/root**, represented by ~; Sobell, page 182).

```
$ su -c 'mv /etc/yum.repos.d/*repo ~'
```

Working with **root** privileges, use an editor to create a file in the **/etc/yum.repos.d** directory named **class.repo** as follows. Substitute the pathname of the mount point displayed in step 3 for **run/media/student/Fedora 19 i386**. Because it contains SPACEs, you must quote the string you assign to **baseurl**.

```
$ cat /etc/yum.repos.d/class.repo
[class-DVD]
name=Class-DVD
baseurl='file:///run/media/student/Fedora 19 i386'
gpgcheck=0
enabled=1
```

b. Working with **root** privileges, give the following command to remove the yum cache.

```
$ su -c 'yum clean all'
```

c. Test that the repository is working by installing the **vim-enhanced** package. Respond with **y** when yum asks you to confirm that you want to install the package.

```
$ su -c 'yum install vim-enhanced'
```

If the repository does not work

tip Confirm that SELinux is in permissive mode by giving the command **setenforce permissive** while working with **root** privileges.

Lab 46: Installing Fedora on the Classroom Server (30–50 minutes)

Objectives

In this lab you will work through the Anaconda installation to install Fedora 19 on the classroom server. This lab continues where Lab 1 left off. This lab is similar to Lab 2 (page 11).

The student system and the classroom server

tip These labs refer to the system the student is working on as the **local system** or the **student system**. They refer to the classroom server system as the **classroom server** or **class**.

Setup

- A VM or physical system that is paused while displaying the Install Image Boot menu (Sobell, Figure 3-4 on page 61).

Reading

Read the following sections:

- "Installing from an Install Image" on page 60 of Sobell
- "Manual/Custom Partitioning" on page 74 of Sobell

Procedure

These steps continue from Lab 1 and assume the Install Image Boot menu (Sobell, Figure 3-4 on page 61) is displayed on the screen and paused. ***Except:*** As you set up the system for installation, name the system **class** instead of **Fedora.XXX**.

If you are using a VM, click on the VM window so the host directs your keystrokes to the VM. Click CONTROL-ALT when you want to direct your keystrokes to the host.

1. Use the ARROW keys to highlight **Test this media & install Fedora 19** and press RETURN to begin the installation. If you have already tested the media you do not need to do so again: Highlight **Install Fedora 19** and press RETURN.

   Ignore the **Press the <ENTER> key to begin the installation process** message; the installation will continue after a moment without intervention.

   Testing the install medium takes a few minutes. After Anaconda completes the test it displays the Welcome to Fedora 19 screen.

2. Highlight the language you want to use during the installation; click **Continue**. After a moment Anaconda displays the Installation Summary screen (Sobell, Figure 3-6 on page 64).

3. Click **Network Configuration** and use the text box labeled **Hostname** to change the hostname to **class.example.com**. Click **Done** at the upper-left corner of the screen to redisplay the Installation Summary screen.
4. Scroll down and click **Installation Destination** to display the Installation Destination screen (Sobell, Figure 3-9 on page 73). The frame labeled **Local Standard Disks** will show a single 20-gigabyte (approximately) disk with a tick in a circle on it. Click **Done** at the upper-left corner of the screen; Anaconda displays the Installation Options window.
5. In the Installation Options window (Sobell, Figure 3-10 on page 74), click the radio button labeled **Automatically configure my Fedora installation ...**, make sure LVM is selected from the drop-down list labeled **Partition scheme**, and click **Continue**. Anaconda redisplays the Installation Summary screen.
6. Click **Begin Installation**; Anaconda displays the Configuration window.

The root password on the classroom server should be a secret

tip The instructor should decide on a password for the **root** user on the classroom server. The instructor should keep that password secret; the students do not need to know the **root** password on the classroom server.

7. See “User Settings” on page 68 of Sobell for instructions on how to add the **root** password (click **Root Password**) and how to add a user with a full name of **Rose Milov**, the username **rose**, and the password **fit714tree** (click **User Creation**). Keep the **root** password secret and do *not* make **rose** an administrator. Do require a password. You will have to click **Done** twice in each window if you specify a weak password. See Sobell, page 136 for information on choosing a secure password.
8. Return to the Configuration window and wait for the installation to complete. Installing Fedora takes a long time; watch the progress bar to see how much of the installation is complete.
9. When Anaconda displays the message near the bottom of the window saying it has successfully installed Fedora and asks you to reboot the system, click **Reboot** and, if you are using a DVD, remove it.

If you are using VMware Player and a DVD and the installation fails at this point

tip If you are installing Fedora from a DVD (and not an ISO image file) using VMware Player and the installation fails at this point, start over from the beginning of Lab 1 (page 7). This time, follow the instructions in the tip on page 9 of this Lab Manual. A failed installation leaves files for the virtual machine on the disk. Either remove this virtual machine before trying to install it again, or give the new virtual machine a different name.

10. The system reboots and displays the Login screen (Sobell, Figure 4-1 on page 91).

11. Log in as Rose; see "Logging In" on page 14 of this Lab Manual.
12. Run gnome-initial-setup; see "Running gnome-initial-setup" on page 15 of this Lab Manual.
13. Follow the instructions to stop blanking the screen in "Stopping GNOME from blanking the screen" on page 17 of this Lab Manual.
14. ***If the classroom server is not connected to the Internet, skip this step.*** See "Two ways to update the installed software" on page 14 of this Lab Manual. Either click the GUI reminder or follow the instructions in step 1 on page 34 of this Lab Manual to update the software on the classroom server. Once the software is updated, make sure to reboot the system and log in again.
15. Set up the classroom server to use a static IP address by following the instructions on page 641 of Sobell. Make sure the address you choose is on the same network as the student systems (which have addresses assigned by DHCP) but that it is outside of the range of addresses assigned by DHCP. Check with the system administrator for more information.

    For example, if DHCP initially assigns the classroom server an IP address of 192.168.206.139 (give the command **ip addr** to display the IP address), you *might* be able to use 192.168.206.250 as a static address for the classroom server. Give the command **route** to display the IP address of the gateway that DHCP assigned. The nameserver line in **/etc/resolv.conf** holds the IP address of the DNS server that DHCP assigned.

Modify the hosts file on all student systems to include the classroom server

tip Lab 18 (page 73) adds the name of the classroom server (**class**) to the **/etc/hosts** file for use in all labs that make use of the classroom server.

If students do not work through Lab 18, you must make sure all student systems have the following line in their **/etc/hosts** file so that they can connect to the classroom server using the hostname **class** instead of an IP address. Replace *classroom-server-IP* with the IP address you assigned to the classroom server in step 15.

```
$ cat /etc/hosts
...
classroom-server-IP class
```

16. Install and start **vsftpd**; put some files in the **/var/ftp/pub** directory (needed for Labs 37 and 38).

    a. Install **vsftpd**.

    ```
    $ su -c 'yum -y install vsftpd'
    ```

    b. Start the **vsftpd** service.

    ```
    $ su -c 'systemctl start vsftpd.service'
    ```

c. Enable the **vsftpd** service so that it starts when the system boots.

```
$ su -c 'systemctl enable vsftpd.service'
```

d. Copy the **/usr/share/themes** directory hierarchy to the **/var/ftp/pub** directory (needed for Labs 39 and 40).

```
$ su -c 'cp -a /usr/share/themes /var/ftp/pub'
```

17. Set up an NFS server, create and populate the **/testing** directory, and export the **/testing** and **/var/ftp/pub** directories (needed for Lab 39).

a. Install the **nfs-utils** package.

```
$ su -c 'yum -y install nfs-utils'
```

b. Make sure **rpcbind** is running.

```
$ systemctl status rpcbind.service
```

c. Use firewall-cmd to open ports TCP 2049 and UDP 2049 in both the running and permanent configurations.

```
$ su -c 'firewall-cmd --add-port=2049/tcp'
$ su -c 'firewall-cmd --permanent --add-port=2049/tcp'
$ su -c 'firewall-cmd --add-port=2049/udp'
$ su -c 'firewall-cmd --permanent --add-port=2049/udp'
```

d. Create a directory named **/testing** and give it 777 permissions.

```
$ su -c 'mkdir /testing'
$ su -c 'chmod 777 /testing'
```

e. Use an editor to create small files named **memo1** and **memo2** in the directory you just created and give the files read-write permission for everyone (666).

```
$ cat /testing/memo1
This file is named memo1.

$ cat /testing/memo2
This file is named memo2.

$ chmod 666 /testing/*
```

f. Working with **root** privileges, use an editor to modify the **/etc/exports** file to share the **/testing** directory with everyone with readonly access and to share the **/var/ftp/pub** directory with everyone with read-write permissions.

```
$ cat /etc/exports
/var/ftp/pub *(ro,sync)
/testing *(rw,sync)
```

g. Use exportfs to export the directory in **/etc/exports** and to verify exported directories.

```
$ su -c 'exportfs -ra'
$ su exportfs
```

h. Set up the **nfs-server.service** service to start each time the system enters multiuser mode and start it now.

```
$ su -c 'systemctl enable nfs-server.service'
$ su -c 'systemctl start nfs-server.service'
```

18. Set up Samba shares (needed by Lab 43).

a. Install the **samba** package.

```
$ su -c 'yum -y install samba'
```

b. Enable and start the **smb** and **nmb** services.

```
$ su -c 'systemctl enable smb.service'
$ su -c 'systemctl enable nmb.service'
$ su -c 'systemctl start smb.service'
$ su -c 'systemctl start nmb.service'
```

c. Set up SELinux so that it allows Samba clients to browse home directories. This command takes a while to complete.

```
$ su -c 'setsebool -P samba_enable_home_dirs on'
```

d. Add a Samba password of **fit714tree** for **rose**.

```
$ su -c 'smbpasswd -a rose'
Password:                    # Enter secret root password
New SMB password: fit714tree
Retype new SMB password: fit714tree
Added user rose.
```

e. Open the firewall so that the Samba shares can be accessed from remote systems.

```
$ su -c 'firewall-cmd --add-port=137/udp'
$ su -c 'firewall-cmd --permanent --add-port=137/udp'
$ su -c 'firewall-cmd --add-port=138/udp'
$ su -c 'firewall-cmd --permanent --add-port=138/udp'
$ su -c 'firewall-cmd --add-port=139/tcp'
$ su -c 'firewall-cmd --permanent --add-port=139/tcp'
$ su -c 'firewall-cmd --add-port=445/tcp'
$ su -c 'firewall-cmd --permanent --add-port=445/tcp'
```

19. Set up a fake printer named **ghost** (needed by Lab 20).

a. Give the following command to display the Print Settings window (Figure 13-1 on page 558 of Sobell).

```
$ su -c system-config-printer
```

b. Click **Add** to configure a new printer. The system-config-printer utility displays the Select Device screen of the New Printer window (Sobell, Figure 13-3 on page 561).

c. When system-config-printer asks if you want to adjust the firewall, click **Adjust Firewall**. This action allows other systems on the local network to see printers you set up on the local system.

d. The system-config-printer utility pauses before it displays a list of devices. Click **Serial Port #1**, leave the default settings, and click **Forward**. After searching for drivers, system-config-printer displays the first Choose Driver screen of the New Printer window (Sobell, Figure 13-5 on page 563).

e. Because no printer is attached to the system, Fedora will not find any drivers. Click **Generic** to select a generic driver from the printer database. Click **Forward**; system-config-printer displays the second Choose Driver screen of the New Printer window (Sobell, Figure 13-6 on page 563).

f. Click **text-only** in the column on the left. Click **Forward**; system-config-printer displays the Describe Printer screen of the New Printer window (Sobell, Figure 13-7 on page 564).

- Replace the text in the text box labeled **Printer Name** with the word **ghost**.
- Replace the text in the text box labeled **Description** with a description of the printer (e.g., **classroom generic printer**).
- Leave the text box labeled **Location** empty.

Click **Apply**; system-config-printer asks if you want to print a test page. Click **Cancel**.

g. On the menubar, click **Server⇨Settings** and put a tick in the check box labeled **Publish shared printers connected to this system**. Click **OK**.

h. Close the Print Settings window.

20. Set up the **sudoers** file to allow Rose on the classroom server to use sudo to work as a privileged user and run mkdir and to mount and unmount certain mount points (needed by Lab 40).

a. Give the following command to open the **/etc/sudoers** file in the vim editor.

```
$ su -c visudo
```

Alternately, you can use an editor of your choice to edit the **/etc/sudoers** file; you must work with **root** privileges.

b. Add the following lines to the bottom of the **sudoers** file.

```
# For the classroom server, allow user rose to mount filesystems for the NFS lab
rose    ALL= /bin/mount, /bin/umount, /usr/bin/mkdir
```

c. Write out the file and exit from the editor.

Deliverables

A newly installed Fedora system that will function as the classroom server.