Chapter 4, Lab 1: The Linux Filesystem (10 Minutes)

Learning Objectives and Outcomes

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 a ordinary file, and mkdir/rmdir to create and remove directory files.

Reading

Read "The Hierarchical Filesystem" through the end of "Directory Files and Ordinary Files" (Sobell, pages 82–87) to gain an understanding of the hierarchical nature of the Linux filesystem. 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 worked only in your home directory. When called by itself, the cd (change directory; Sobell, page 92) command makes your home directory the working directory (the directory you are working in). The pwd (print working directory; Sobell, page 86) 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?
2. The mkdir (make directory; Sobell, page 90) 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 is a directory.
3. An absolute pathname (Sobell, page 88) 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.
4. Use vim to create a file named **fox** in the **two** directory. Use ls to list the name of **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.

6. Make your home directory the working directory. A relative pathname (Sobell, page 89) 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.
7. The rmdir utility removes an empty directory. Show that rmdir cannot remove the **two** directory while it holds a file.

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

Deliverables

This lab gives you practice using vim, cd, pwd, rm, mkdir, and rmdir. When you use script to capture your work in the lab, you can turn the resulting file in to your instructor.

Chapter 4, Lab 2: File Access Permissions (10 Minutes)

Learning Objectives and Outcomes

In this lab you will learn to use touch to create an empty file, **ls –l** to display file ownership and permissions, and chmod to change file permissions.

Reading

Read about access permissions (Sobell, page 98) through the section on chmod (Sobell, page 101).

Procedure

1. You can use the touch utility (Sobell, pages 94 and 985) to create an empty file quickly. Use touch to create a file named **dog** in the working directory. Use ls to confirm the file was created.

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

2. Use **ls –l** (Sobell, page 98) to display the permissions of the file you just created. Who owns the file? To which group does it belong? Which permissions does the owner of the file have? The group? Others?
3. Display the permissions of **/bin/bash**. Who owns the file? To which group does it belong? Which permissions does the owner of the file have? The group? Others? Which permissions apply to you?
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 100), 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.

Deliverables

This lab gives you practice using **ls –l**, touch, and chmod. When you use script to capture your work in the lab, you can turn the resulting file in to your instructor.