Network Administration Lab
Experiment#2
File System Management

Dr. Ahmed Awad

September 22, 2020

1 Objectives

The purpose of this experiment is to practice the basic administrative tasks for file management within system administration. This includes understanding different file system permissions, types of files, sticky bits, and mounting

2 Overview

One of the primary tasks of a network and system administrator lies within file system management. As different users of an organization access their file systems, setting proper permissions for different files preserves information confidentiality.
A file system organizes a system storage resources. It logically consists of four main components:

a. Name space: A way to name the objects and organize them in a hierarchy.

b. API: System calls for manipulating the objects.

c. Security model: Scheme for protecting, hiding, and sharing objects either locally or remotely.

d. Implementation: A software to tie logical models to hardware.

A file system in Linux typically has a tree structure that starts from the root. Mapping a directory within the existing file system tree to the root of a newly attached file system is called **Mounting**. Mounting should be executed to allow accessing other file systems (such as external driver files, or network file systems).
One of the important tasks of a network administrator is to manage file permissions through properly sitting the permission bits. This includes setting read, write, and execution permissions for the file owner, the users in the owner group, and all other users.

3 Procedure

3.1 File System Partitions

a. A hard disk is divided into partitions. **For what file systems can those partitions be assigned?**

b. Use **fdisk -l** command to find the partition under which Linux file system has been assigned on your machine.

c. Use the command **mount** to check which partitions are already mounted and where they are mounted. **What type of file system does the directory /proc have?**

d. Check the file systems that are supported by your Linux operating system using the information provided under **/proc/filesystems**.

e. When booting Linux, a set of file systems are automatically mounted. Check the lists of automatically mountable files in your machine through checking the content of the file **/etc/fstab**. **Explain what does each field mean in this file?**

f. You have to mount the Windows file system so that it can be accessed through your Linux machine. For this purpose, do the following steps:

1. Decide on which partition does the Windows file system exist. **How can you do that?**
2. Append the following line to the file **/etc/fstab** where XYZ represents the partition that has been assigned to the Windows file system:
   /dev/XYZ /mnt/win vfat noauto 0 0
   What does each of the following options mean: -vfat, noauto?
3. Create the directory **/mnt/win**. **What command do you use for this purpose? Do you need root permissions? Why?**
4. Mount the Window file system into your Linux file system using the command **mount /mnt/win**
5. Check the result and check syslog messages in case of failures.
6. Remove the extra line into /etc/fstab.
7. Remove the folder win under /mnt directory

3.2 File Types

a. What are the types of files in Linux file system?

b. Create a directory named **Exp5**. In this directory create a file named **file1.txt**.

c. What is the absolute path of the file **file1.txt**?

d. List the different files/folders under the directory /dev. From this list, provide an example for each of the following:

   1. A regular file.
   2. A directory
   3. A character device.
   4. A block device
   5. A symbolic link.

e. What is the difference between a block device file and a character device file?

f. Write a bash shell script that receives the absolute path for a directory as an argument from the command line. Then it performs each of the following:

   1. Checks whether the directory exists or not. If does not exist, it generates an error message.
   2. If exists, it finds all the symbolic link files under this directory.
   3. Creates a file named **Links.txt** in which it prints the names of all the symbolic link files under that directory.

g. Create a socket file as follows:

   1. Install the **socket** package.
   2. Use the manual page of socket command to create a UNIX domain stream socket.
   3. Use port number 3425.
   4. Check all listening sockets and make sure that your socket created socket is there.
   5. Kill your running socket process.

3.3 File Symbolic Links

a. What is the key difference between a soft link and a hard link?

b. Create two files named **f1** and **f2**.

c. Write "This is the first file" in file f1.

d. Write "This is the second file" in file f2.

e. Create a hard link for file f1 named **f1-hard**.

f. Create a soft link for file f2 named **f2-soft**.

g. Make sure that both links have been successfully created.

h. Rename file1 to file1-new.

i. Rename file2 to file2-new.

j. Print the content of both the soft-link and the hard-link and state your conclusions.

k. Write a bash shell script that receives the absolute path of a soft link then it performs the following:

   1. Confirms that the soft link exists.
   2. If not, generates an error message.
   3. If exists, your script should print the absolute path of the file for that the symbolic link points.

3.4 File Permissions

a. Explain the file permission bits in Linux?

b. Explain all the fields when executing the command **ls -l**

c. Create a file and set its permissions to be read and write for all users while only executable for the owner of the file.

d. What are the types of executable files?

e. Write a bash shell script that receives the absolute path of a directory and then prints the names of all the executable files by the owner of those files.

f. Change the ownership of a file so that the root becomes its owner.

g. Provide a case in which changing a file ownership is necessary?

3.5 File Naming

a. Create a file whose name contains a space. For example, **My File.txt**. **How can you do that?**

b. Create a file whose name starts with a dash. For example, **-foo**. **How can you do that?**

c. Try to remove the above created files. **Explain how can you do that?**

3.6 Sticky Bit

a. What is a sticky bit?

b. Check the permissions of the directories under the root directory. Check carefully the permission bits of the directory **/tmp**. **What do you notice?**

c. Write a bash shell script that sets the sticky bit of an input directory.