Course: Operating Systems

Instructor: Dr. Raheel Ahmed Memon

Lab-4

Objectives

1. Manipulation of users and groups in Linux (add, remove, move and edit)
2. File Permissions and Ownership
3. Self-Learning and Sharing

Manipulation of Users and Groups in Linux

Important files

`/etc/passwd`

`/etc/groups`

`/etc/shadow`

To see the existing users:

```
$ cat /etc/passwd
```

To add a new users:

```
$ adduser raheel
```

OR

```
$ useradd raheel
```

Then check if user is added or not

```
$ cat /etc/passwd
```

…

…

```
raheel:x:500:500: raheel ahmed memon, 7, , /home/raheel :/bin/bash
```

Task # 01

The default entries can be edited using by editing `/etc/passwd` file, for example, change rahee user's default directory to `/home/raheel/Desktop`

Change user:

```
$ su – Raheel
```

See currently login user

```
$ whoami
```

User id can be observed using `id` command, it returns the current login id, in case of raheel it is 500, while root is always 0.

Changing the password:

```
$ passwd raheel
New Password: *******
```

Change owner/user and group

For Group:

```
$ chown :raheel file1
```

For User:

```
$ chown raheel file1
```

File Permissions and Ownership

All Linux files and directories have ownership and permissions. You can change permissions, and sometimes ownership, to provide greater or lesser access to your files and directories. File permissions also determine whether a file can be executed as a command.
If you type ls -l or dir, you see entries that look like this:

```
-rw-r—r— 1 fido users 163 Dec 7 14:31 myfile
```

The -rw-r—r— represents the permissions for the file myfile. The file's ownership includes fido as the owner and users as the group.

File and Directory Ownership
When you create a file, you are that file's owner. Being the file's owner gives you the privilege of changing the file's permissions or ownership. Of course, once you change the ownership to another user, you can't change the ownership or permissions anymore!
File owners are set up by the system during installation. Linux system files are owned by IDs such as root, uucp, and bin. Do not change the ownership of these files.

The chown command
Use the chown (change ownership) command to change ownership of a file. The syntax is chown `<owner> <filename>`. In the following example, you change the ownership of the file myfile to root:

```
$ ls -l myfile
```

```
-rw-r—r— 1 fido users 114 Dec 7 14:31 myfile

$ chown root myfile

$ ls -l myfile

-rw-r—r— 1 root users 114 Dec 7 14:31 myfile
```

To make any further changes to the file myfile, or to chown it back to fido, you must use su or log in as root.

File Permissions

Linux lets you specify read, write, and execute permissions for each of the following: the owner, the group, and "others" (everyone else).

read permission enables you to look at the file. In the case of a directory, it lets you list the directory's contents using ls.

write permission enables you to modify (or delete!) the file. In the case of a directory, you must have write permission in order to create, move, or delete files in that directory.

execute permission enables you to execute the file by typing its name. With directories, execute permission enables you to cd into them.

For a concrete example, let's look at myfile again:

```
-rw-r—r— 1 fido users 163 Dec 7 14:31 myfile
```

The first character of the permissions is -, which indicates that it's an ordinary file. If this were a directory, the first character would be d.

The next nine characters are broken into three groups of three, giving permissions for owner, group, and other. Each triplet gives read, write, and execute permissions, always in that order. Permission to read is signified by an r in the first position, permission to write is shown by a w in the second position, and permission to execute is shown by an x in the third position. If the particular permission is absent, its space is filled by -.

In the case of myfile, the owner has rw-, which means read and write permissions. This file can't be executed by typing myfile at the Linux prompt.

The group permissions are r—, which means that members of the group "users" (by default, all ordinary users on the system) can read the file but not change it or execute it.

Likewise, the permissions for all others are r—: read-only.

File permissions are often given as a three-digit number—for instance, 751. It's important to understand how the numbering system works, because these numbers are used to change a file's permissions. Also, error messages that involve permissions use these numbers.

The first digit codes permissions for the owner, the second digit codes permissions for the group, and the third digit codes permissions for other (everyone else).

The individual digits are encoded by summing up all the "allowed" permissions for that particular user as follows:

Read permission	4
write permission	2
execute permission	1

Therefore, a file permission of 751 means that the owner has read, write, and execute permission (4+2+1=7), the group has read and execute permission (4+1=5), and others have execute permission (1).

If you play with the numbers, you quickly see that the permission digits can range between 0 and 7, and that for each digit in that range there's only one possible combination of read, write, and execute permissions.

Task-2

Changing File Permissions

To change file permissions, use the **chmod** (change [file] mode) command. The syntax is

chmod <specification> file

There are two ways to write the permission specification. One is by using the numeric coding system for permissions:

Suppose that you are in the home directory.

```
$ ls -l myfile

-rw-r—r— 1 fido users 114 Dec 7 14:31 myfile

$ chmod 345 myfile

$ ls -l myfile

—wxr—r-x 1 fido users 114 Dec 7 14:31 myfile

$ chmod 701 myfile

$ ls -l myfile

-rwx——x 1 root users 114 Dec 7 14:31 myfile
```

This method has the advantage of specifying the permissions in an absolute, rather than relative, fashion. Also, it's easier to tell someone "Change permissions on the file to seven-five-five" than to say "Change permissions on the file to read-write-execute, read-execute, read-execute."

You can also use letter codes to change the existing permissions. To specify which of the permissions to change, type u (user), g (group), o (other), or a (all). This is followed by a + to add permissions or a - to remove them. This in turn is followed by the permissions to be added or removed. For example, to add execute permissions for the group and others, you would type:

```
$ chmod go+r myfile
```

Other ways of using the symbolic file permissions are described in the chmod man page.

Task - 3

Create a c file and change permission for owner to write only. So that when he triggers the command to execute he receives the permission issue.

Self-Learning and Sharing

Explore the methods to Add Multiple Users