Mawlana Bhashani Science and Technology University

Lab-Report

Report No: 04

Course code: ICT-3110

Course title: Operating Systems Lab

Date of Performance :

Date of Submission : 30/09/2020

Submitted by

Name: Md Humayun Kabir

ID:IT-18026

3rd year 1stsemester

Session: 2017-2018

Dept. of ICT

MBSTU.

Submitted To

Nazrul Islam

Assistant Professor

Dept. of ICT

MBSTU.

Experiment No : 04

Experiment Name : File Operation and Permission.

Thoery :

In this lab,we will discuss in detail about file permission and access modes in Linux. File ownership is an important component of Linux that provides a secure method for storing files. Every file in Linux has the following attributes −

- **Owner permissions** − The owner's permissions determine what actions the owner of the file can perform on the file.
- **Group permissions** − The group's permissions determine what actions a user, who is a member of the group that a file belongs to, can perform on the file.
- **Other (world) permissions** − The permissions for others indicate what action all other users can perform on the file.

The Permission Indicators

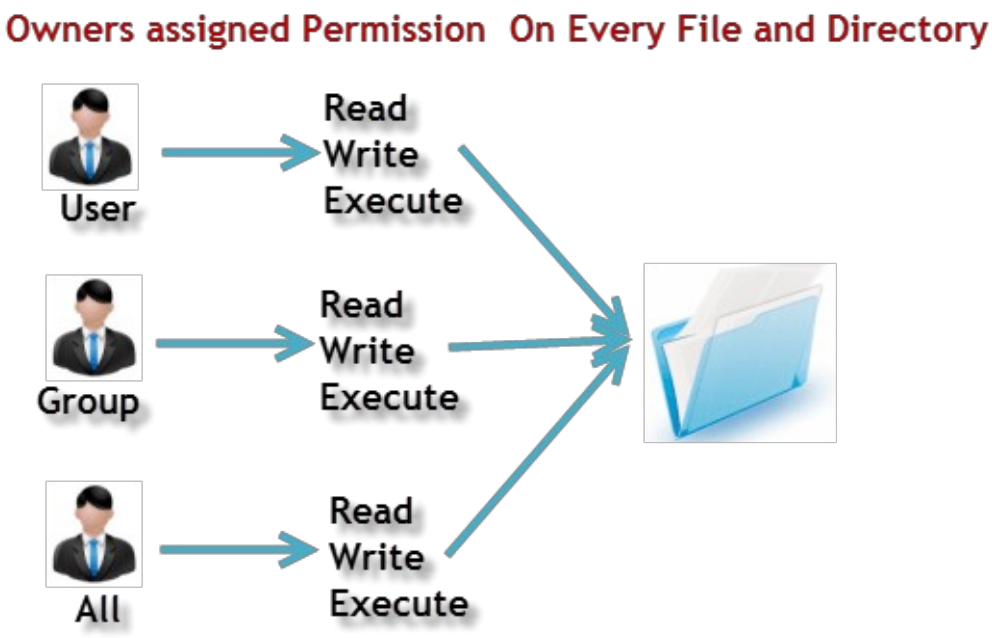
While using **ls -l** command, it displays various information related to file permission as follows −

```
$ls -l /home/Desktop
-rwxr-xr--  1 Desktop   users 1024  Nov 2 00:10  myfile
drwxr-xr--- 1 Desktop   users 1024  Nov 2 00:10  mydir
```

Here, the first column represents different access modes, i.e., the permission associated with a file or a directory.

The permissions are broken into groups of threes, and each position in the group denotes a specific permission, in this order: read (r), write (w), execute (x) −

- **Read:** This permission give you the authority to open and read a file. Read permission on a directory gives you the ability to lists its content.
- **Write:** The write permission gives you the authority to modify the contents of a file. The write permission on a directory gives you the authority to add, remove and rename files stored in the directory. Consider a scenario where you have to write permission on file but do not have write permission on the directory where the file is stored. You will be able to modify the file contents. But you will not be able to rename, move or remove the file from the directory.
- **Execute:** In Windows, an executable program usually has an extension ".exe" and which you can easily run. In Unix/Linux, you cannot run a program unless the execute permission is set. If the execute permission is not set, you might still be able to see/modify the program code(provided read & write permissions are set), but not run it.



In this lab, we will discuss in detail about file operation in Linux. All data in Linux is organized into files. All files are organized into directories. These directories are organized into a tree-like structure called the filesystem.

When you work with Unix/Linux, one way or another, you spend most of your time working with files. This will help us understand how to create and remove files, copy and rename them, create links to them, etc.

In Unix/Linux, there are three basic types of files –

- **Ordinary Files** – An ordinary file is a file on the system that contains data, text, or program instructions. In this tutorial, you look at working with ordinary files.
- **Directories** – Directories store both special and ordinary files. For users familiar with Windows or Mac OS, Unix directories are equivalent to folders.
- **Special Files** – Some special files provide access to hardware such as hard drives, CD-ROM drives, modems, and Ethernet adapters. Other special files are similar to aliases or shortcuts and enable you to access a single file using different names.

Working Procedure :

Implementation of file operation –

1. **Listing Files:**

ls -l :

```
humayun@HK:~$ ls -l
total 57720
drwxr-xr-x 4 humayun humayun     4096 Sep 30 07:51  Desktop
drwxr-xr-x 6 humayun humayun     4096 Sep 25 17:52  Documents
drwxr-xr-x 9 humayun humayun    12288 Sep 16 23:16  Downloads
-rw-r--r-- 1 humayun humayun     8980 Feb 25  2020  examples.desktop
-rw-rw-r-- 1 humayun humayun   921891 Feb 25  2020  Firefox_wallpaper.png
drwxr-xr-x 3 humayun humayun     4096 Jul 17 09:35  hasib
-rw-r--r-- 1 humayun humayun        0 Mar 10  2020  hellow.txt
-rw-r--r-- 1 humayun humayun   423297 Jun 15 14:41  IT-18026-CT-01.pdf.ps
```

Here is the information about all the listed columns −

- **First Column** − Represents the file type and the permission given on the file. Below is the description of all type of files.
- **Second Column** − Represents the number of memory blocks taken by the file or directory.
- **Third Column** − Represents the owner of the file. This is the Unix user who created this file.
- **Fourth Column** − Represents the group of the owner. Every Unix user will have an associated group.
- **Fifth Column** − Represents the file size in bytes.
- **Sixth Column** − Represents the date and the time when this file was created or modified for the last time.
- **Seventh Column** − Represents the file or the directory name.

2.Hidden Files :

An invisible file is one, the first character of which is the dot or the period character (.). Unix programs (including the shell) use most of these files to store configuration information.

Some common examples of the hidden files include the files −

- **.profile** − The Bourne shell (sh) initialization script
- **.kshrc** − The Korn shell (ksh) initialization script
- **.cshrc** − The C shell (csh) initialization script
- **.rhosts** − The remote shell configuration file

To list the invisible files, specify the **-a** option to **ls** −

- **Single dot (.)** – This represents the current directory.
- **Double dot (..)** – This represents the parent directory.

3.Creating Files :

We can use the **vi** editor to create ordinary files on any Linux/Unix system. We simply need to give the following command –

```
humayun@HK:~$ vi new.txt
```

The above command will open a file with the given filename. Now,we have to press the key **i** to come into the edit mode. Once we are in the edit mode, we can start writing our content in the file as in the following program –

```
Hard time gone through……
```

Once you are done with the program, follow these steps –

- Press the key **esc** to come out of the edit mode.
- Press two keys **Shift + ZZ** together to come out of the file completely.

You will now have a file created with **filename** in the current directory.

4.Editing Files :

We can edit an existing file using the **vi** editor. We will discuss in short how to open an existing file –

$ vi filename

```
humayun@HK:~$ vi new.txt
```

Once the file is opened, we can come in the edit mode by pressing the key **i** and then we can proceed by editing the file. If we want to move here and there inside a file, then first we need to come out of the edit mode by pressing the key **Esc**. After this, we can use the following keys to move inside a file –

- **l** key to move to the right side.
- **h** key to move to the left side.

- **k** key to move upside in the file.
- **j** key to move downside in the file.

So using the above keys, we can position our cursor wherever we want to edit. Once we are positioned, then we can use the **i** key to come in the edit mode. Once we are done with the editing in our file, press **Esc** and finally two keys **Shift + ZZ** together to come out of the file completely.

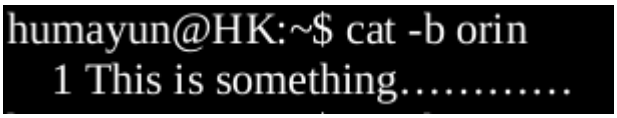
5. Display Content of a File :

We can use the **cat** command to see the content of a file. Following is a simple example to see the content of the above created file –

```
$ cat filename
This is unix file....I created it for the first time.....
I'm going to save this content in this file.
$
```

We can display the line numbers by using the **-b** option along with the **cat** command as follows –

```
$ cat -b filename
1  This is unix file....I created it for the first time.....
2  I'm going to save this content in this file.
$
```



6.Counting Words in a File :

We can use the **wc** command to get a count of the total number of lines, words, and characters contained in a file. Following is a simple example to see the information about the file created above –

```
$ wc filename
```

2 19 103 filename
$

```
humayun@HK:~$ cat -b orin
1 5 29 orin
```

Here is the detail of all the four columns −

- **First Column** − Represents the total number of lines in the file.
- **Second Column** − Represents the total number of words in the file.
- **Third Column** − Represents the total number of bytes in the file. This is the actual size of the file.
- **Fourth Column** − Represents the file name.

We can give multiple files and get information about those files at a time. Following is simple syntax −

```
$ wc filename1 filename2 filename3
```

7.Copying Files :

To make a copy of a file use the **cp** command. The basic syntax of the command is −

```
$ cp source_file destination_file
```

Following is the example to create a copy of the existing file **filename**.

```
$ cp filename copyfile
$
```

We will now find one more file **copyfile** in our current directory. This file will exactly be the same as the original file **filename**.

8.Renaming Files :

To change the name of a file, use the **mv** command. Following is the basic syntax −

```
$ mv old_file new_file
```

The following program will rename the existing file **filename** to **newfile**.

```
$ mv filename newfile
$
```

The **mv** command will move the existing file completely into the new file. In this case, we will find only **newfile** in our current directory.

9.Deleting Files :

To delete an existing file, use the **rm** command. Following is the basic syntax −

```
$ rm filename
```

Caution − A file may contain useful information. It is always recommended to be careful while using this **Delete** command. It is better to use the **-i** option along with **rm** command.

Following is the example which shows how to completely remove the existing file **filename**.

```
$ rm filename
$
```

We can remove multiple files at a time with the command given below −

```
$ rm filename1 filename2 filename3
$
```

File permission :

1.Viweing Permission :

Use the **ls** command's -l option to view the permissions (or **file mode**) set for the contents of a directory, for example:

```
humayun@HK:~$ ls -l
total 57720
drwxr-xr-x 4 humayun humayun     4096 Sep 30 07:51  Desktop
drwxr-xr-x 6 humayun humayun     4096 Sep 25 17:52  Documents
drwxr-xr-x 9 humayun humayun    12288 Sep 16 23:16  Downloads
-rw-r--r-- 1 humayun humayun     8980 Feb 25  2020  examples.desktop
-rw-rw-r-- 1 humayun humayun   921891 Feb 25  2020  Firefox_wallpaper.png
drwxr-xr-x 3 humayun humayun     4096 Jul 17 09:35  hasib
-rw-r--r-- 1 humayun humayun        0 Mar 10  2020  hellow.txt
-rw-r--r-- 1 humayun humayun   423297 Jun 15 14:41  IT-18026-CT-01.pdf.ps
```

2.Changing permissions :

chmod is a command in Linux and other Unix-like operating systems that allows to change the permissions (or access mode) of a file or directory.

Text method

To change the permissions — or access mode — of a file, use the chmod command in a terminal. Below is the command's general structure:

chmod **who**=permissions filename

Where who is any from a range of letters, each signifying who is being given the permission. They are as follows:

- u: the **user** that owns the file.
- g: the **user group** that the file belongs to.
- o: the **o**ther users, i.e. everyone else.
- a: all of the above; use this instead of typing ugo.

The permissions are the same as discussed in **Viewing permissions** (r, w amd x).

Now have a look at some examples using this command. Suppose you became very protective of the Documents directory and wanted to deny everybody but yourself, permissions to read, write, and execute (or in this case search/look) in it:

Before: drwxr-xr-x 6 archie users 4096 Jul 5 17:37 Documents

$ chmod g= Documents

$ chmod o= Documents

3.Copying permissions :

It is possible to tell chmod to copy the permissions from one class, say the owner, and give those same permissions to group or even all. To do this, instead of putting r, w or x after the =, put another **who** letter. e.g:

Before: -rw-r--r-- 1 archie users 5120 Jun 27 08:28 foobar

$ chmod g=u foobar

After: -rw-rw-r-- 1 archie users 5120 Jun 27 08:28 foobar

This command essentially translates to "change the permissions of group (g=), to be the same as the owning user (=u). Note that you cannot copy a set of permissions as well as grant new ones e.g.:

$ chmod g=wu foobar

In that case *chmod* throw an error.

4.Numeric Method :

chmod can also set permissions using numbers.

Using numbers is another method which allows you to edit the permissions for all three owner, group, and others at the same time, as well as the setuid, setgid, and sticky bits. This basic structure of the code is this:

$ chmod xxx filename

Where xxx is a 3-digit number where each digit can be anything from 0 to 7. The first digit applies to permissions for owner, the second digit applies to permissions for group, and the third digit applies to permissions for all others.

In this number notation, the values r, w and x have their own number value:

r=4

w=2

x=1

To come up with a 3-digit number you need to consider what permissions you want owner, group, and all others to have, and then total their values up. For example, if you want to grant the owner of a directory read write and execution permissions, and you want group and everyone else to have just read and execute permissions, you would come up with the numerical values like so:

- Owner:rwx=4+2+1=7
- Group:r-x=4+0+1=5

- Other:r-x=4+0+1=5

$ chmod 755 *filename*

Discussion :

This lab helps us to access file operations and permissions. We have seen that how file operation and permission can happen in Linux Operating System. In future we can do such operations using Linux Operating System.