



Mawlana Bhashani Science and Technology University

Lab-Report

Report No: 04

Course code: ICT-3110

Course title: Operating System Lab

Date of Performance: 20-09-2020

Date of Submission:

Submitted by

Name: Jannatul Ferdush Dhina

ID:IT-18012

3rd year 1st semester

Session: 2014-2015

Dept. of ICT

MBSTU.

Submitted To

Nazrul Islam

Assistant Professor

Dept. of ICT

MBSTU.

Experiment No: 04

Experiment Name: File Operation and Permission.

Theory:

In this lab, we will discuss in detail about file permission and access modes in Unix. File ownership is an important component of Unix that provides a secure method for storing files. Every file in Unix 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/Jannatul Dhina
-rwxr-xr-- 1Jannatul Dhina  users 1024  Nov 2 00:10  myfile
drwxr-xr--- 1 1Jannatul Dhina 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 of a file are the first line of defense in the security of a Unix system. The basic building blocks of Unix permissions are the **read**, **write**, and **execute** permissions, which have been described below –

Read:

Grants the capability to read, i.e., view the contents of the file.

Write:

Grants the capability to modify, or remove the content of the file.

Execute:

User with execute permissions can run a file as a program.

Owner assigned permission:

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.

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.

1.Listing Files

To list the files and directories stored in the current directory, use the following command –

```
$ls
```

Here is the sample output of the above command –

```
$ls
```

```
bin      hosts lib    res.03
ch07     hw1  pub    test_results
ch07.bak hw2   res.01 users
docs     hw3   res.02 work
```

- 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.

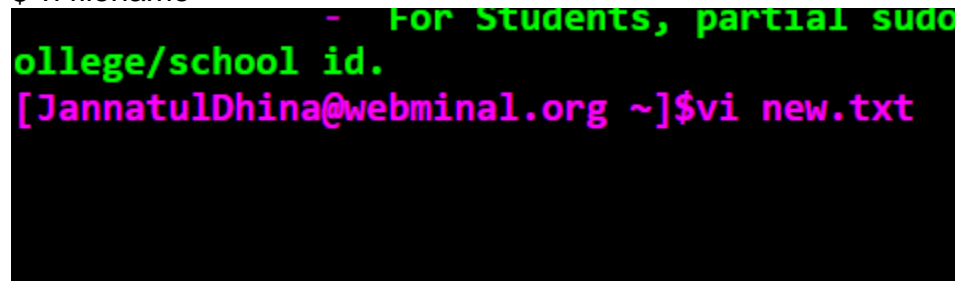
Some common examples of the hidden files include the files –

- .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

3.Creating Files

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

\$ vi filename



4.Editing Files

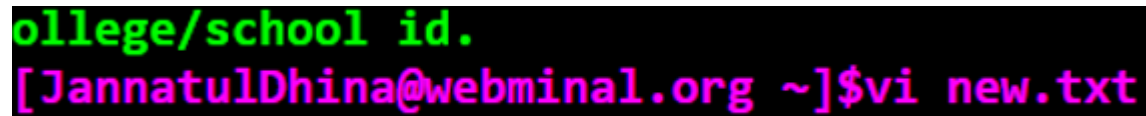
You can edit an existing file using the **vi** editor. We will discuss in short how to open an existing file –

\$ vi filename

Once the file is opened, we can come in the edit mode by pressing the key **i** and then you can proceed by editing the file. If you want to move here and there inside a file, then first you need to come out of the edit mode by pressing the key **Esc**. After this, you 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, you can position your cursor wherever you want to edit. Once you are positioned, then you can use the **i** key to come in the edit mode. Once you are done with the editing in your file, press **Esc** and finally two keys **Shift + ZZ** together to come out of the file completely.

A terminal window with a black background. The first line is 'ollege/school id.' in green text. The second line is '[JannatulDhina@webminal.org ~]\$vi new.txt' in pink text.

```
ollege/school id.  
[JannatulDhina@webminal.org ~]$vi new.txt
```

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  
$
```

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 your 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 your 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  
$
```

You can remove multiple files at a time with the command given below –

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

File Permission:

1. Viewing Permission: Use the ls command's -l option to view the permissions (or file mode) set for the contents of a directory.

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 other 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 and 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
```

Discussion : This lab was about access file operations and permissions. Now it is possible for us to do such operation in LINUX.