Experiment No. 1

Title: Introduction to different Operating Systems and study file permissions in Linux

Batch: B4 **Roll No: 16010420117** **Experiment No: 1**

Aim: Introduction to Operating Systems and implementation of file permission commands in Linux.

Resources needed: Any open source OS/ online CoCalc editor, internet

Theory:

Pre lab/Prior concepts:

Introduction of Operating System

An operating system acts as an intermediary between the user of a computer and computer hardware. The purpose of an operating system is to provide an environment in which a user can execute programs in a convenient and efficient manner.

An operating system is system software that manages the computer hardware. The hardware must provide appropriate mechanisms to ensure the correct operation of the computer system and to prevent user programs from interfering with the proper operation of the system.

Types of Operating System

Batch Operating System- Sequence of jobs in a program on a computer without manual interventions.

Time sharing operating System- allows many users to share the computer resources.(Max utilization of the resources).

Distributed operating System- Manages a group of different computers and makes appear to be a single computer.

Network operating system- computers running in different operating system can participate in common network (It is used for security purpose).

Real time operating system – meant applications to fix the deadlines.

Linux Distribution

Linux distribution is an operating system that is made up of a collection of software based on Linux kernel or you can say distribution contains the Linux kernel and supporting libraries and software. And you can get Linux based operating system by downloading one of the Linux distributions and these distributions are available for different types of devices like embedded devices, personal computers, etc. Around 600 + Linux Distributions are available and some of the popular Linux distributions are:

MX Linux, Manjaro, Linux Mint, elementary, Ubuntu, Debian, Solus, Fedora, openSUSE, Deepin

Ownership of Linux files

Every file and directory on your Unix/Linux system is assigned 3 types of owner, given below.

User

A user is the owner of the file. By default, the person who created a file becomes its owner. Hence, a user is also sometimes called an owner.

Group

A user- group can contain multiple users. All users belonging to a group will have the same access permissions to the file. Suppose you have a project where a number of people require access to a file. Instead of manually assigning permissions to each user, you could add all users to a group, and assign group permission to file such that only this group members and no one else can read or modify the files.

Other

Any other user who has access to a file. This person has neither created the file, nor does he belong to a usergroup who could own the file. Practically, it means everybody else. Hence, when you set the permission for others, it is also referred as set permissions for the world.

Now, the big question arises how does Linux distinguish between these three user types.

Let us understand the Permission system on Linux.

Permissions

Every file and directory in your Linux system has following 3 permissions defined for all the 3 owners discussed above.

Read: This permission gives you the authority to open and read a file. Read permission on a directory gives you the ability to lists it’s 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 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.

Access Permissions of files

Read access (mnemonic: r, binary weight: 4)

Permission to read the file; for directories this means the permission to list the contents of the file.

Write access (mnemonic: w, binary weight: 2)

Permission to modify the file; for directories, this means the permission to create or delete files.

Execute access (mnemonic: x, binary weight: 1)

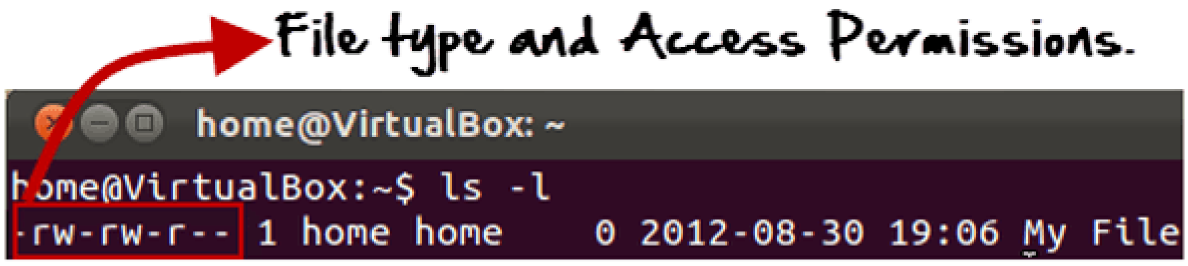
Permission to execute the file; for directories, this means the permission to access files in the directory.

In the above output, these permissions appear in groups of three, discarding the very first character. Thus each triad is made of ``rwx" and the absence of a permission is marked by the dash ``-". The three triads represent permissions for the file owner, the file group and the other groups respectively. The following interpretations can be thus be made about the file: The file owner ``sameer" has read and write permissions.
The file group ``users" has only read permissions. The
other groups have only read permissions.

Owners can define the permissions on every file and folder.

ls - l on terminal gives ***ls***

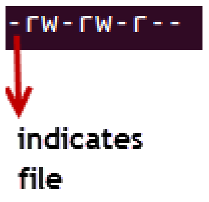
- l



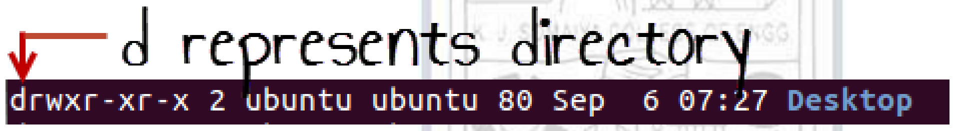
```
home@VirtualBox:~$ ls -l
-rw-rw-r-- 1 home home     0 2012-08-30 19:06 My File
```

Here, we have highlighted **'-rw-rw-r--'**and this weird looking code is the one that tells us about the permissions given to the owner, user group and the world.

Here, the first '**-**' implies that we have selected a file.p>



Else, if it were a directory, **d** would have been shown.



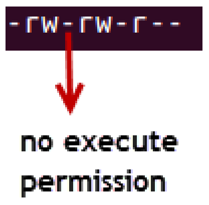
```
drwxr-xr-x 2 ubuntu ubuntu 80 Sep  6 07:27 Desktop
```

The characters are pretty easy to remember.

r = read permission
w = write permission
x = execute permission
- = no permission

Let us look at it this way.

The first part of the code is **'rw-'**. This suggests that the owner 'Home' can:



- Read the file
- Write or edit the file
- He cannot execute the file since the execute bit is set to '-'.

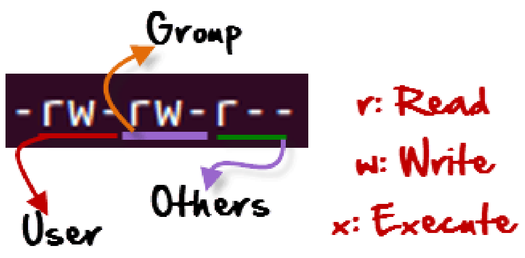
By design, many Linux distributions like Fedora, CentOS, Ubuntu, etc. will add users to a group of the same group name as the user name. Thus, a user 'tom' is added to a group named 'tom'.

The second part is **'rw-'.** It for the user group 'Home' and group-members can:

- Read the file
- Write or edit the file

The third part is for the world which means any user. It says **'r--'.** This means the user can only:

Read the file



Changing file/directory permissions with 'chmod' command

Say you do not want your colleague to see your personal images. This can be achieved by changing file permissions.

We can use the '**chmod'** command which stands for 'change mode'. Using the command, we can set permissions (read, write, execute) on a file/directory for the owner, group and the world.

Syntax: chmod permissions filename

There are 2 ways to use the command -

1. **Absolute mode**
2. **Symbolic mode**

Absolute (Numeric) Mode

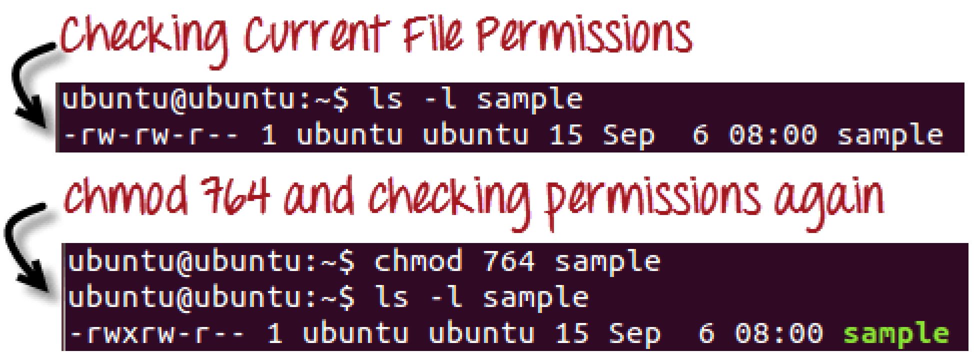
In this mode, file permissions are not represented as characters but a three-digit octal number.

The table below gives numbers for all for permissions types.

Number	**Permission Type**	**Symbol**
0	No Permission	---
1	Execute	--x
2	Write	-w-
3	Execute + Write	-wx
4		r--
5	Read + Execute	r-x

6	Read +Write	rw-
7	Read + Write +Execute	rwx

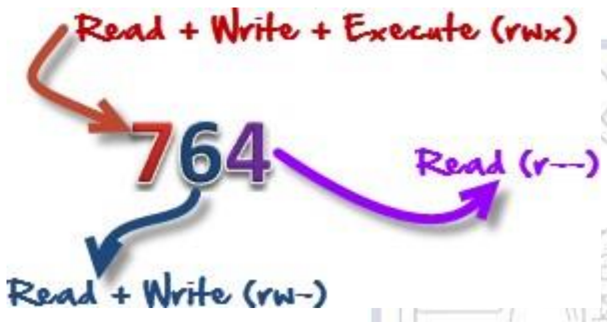
Let's see the chmod command in action.


Checking Current File Permissions
chmod 764 and checking permissions again

```
ubuntu@ubuntu:~$ ls -l sample
-rw-rw-r-- 1 ubuntu ubuntu 15 Sep  6 08:00 sample
```

```
ubuntu@ubuntu:~$ chmod 764 sample
ubuntu@ubuntu:~$ ls -l sample
-rwxrw-r-- 1 ubuntu ubuntu 15 Sep  6 08:00 sample
```

In the above-given terminal window, we have changed the permissions of the file 'sample to '764'.



'764' absolute code says the following:

- Owner can read, write and execute
- Usergroup can read and write
- World can only read

This is shown as '-rwxrw-r-

This is how you can change the permissions on file by assigning an absolute number.

Symbolic Mode

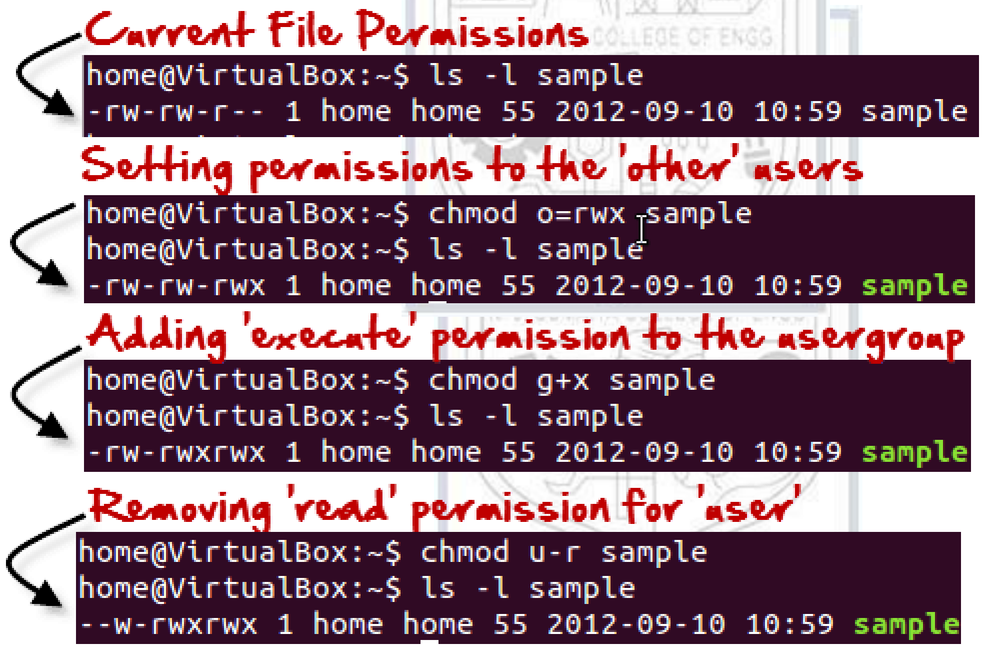
In the Absolute mode, you change permissions for all 3 owners. In the symbolic mode, you can modify permissions of a specific owner. It makes use of mathematical symbols to modify the file permissions.

Operator	**Description**
+	Adds a permission to a file or directory
-	Removes the permission
=	Sets the permission and overrides the permissions set earlier.

The various owners are represented as:

User Denotations	
u	user/owner
g	group
o	other
a	all

We will not be using permissions in numbers like 755 but characters like rwx. Let's look into an example



```
home@VirtualBox:~$ ls -l sample
-rw-rw-r-- 1 home home 55 2012-09-10 10:59 sample
```

Setting permissions to the 'other' users

```
home@VirtualBox:~$ chmod o=rwx sample
home@VirtualBox:~$ ls -l sample
-rw-rw-rwx 1 home home 55 2012-09-10 10:59 sample
```

Adding 'execute' permission to the usergroup

```
home@VirtualBox:~$ chmod g+x sample
home@VirtualBox:~$ ls -l sample
-rw-rwxrwx 1 home home 55 2012-09-10 10:59 sample
```

Removing 'read' permission for 'user'

```
home@VirtualBox:~$ chmod u-r sample
home@VirtualBox:~$ ls -l sample
--w-rwxrwx 1 home home 55 2012-09-10 10:59 sample
```

Activities:

1. Students have to explore different types of operating systems and differentiate them on the basis of pros and cons.
2. Take any 5 domains like gaming, finance, banking etc and suggest which OS will be best suitable for these domains.

<u>Results: No snapshots must be taken</u>

Students should log the results of these commands in a separated file(*.txt/doc/docx)

and upload it on the link shared by the faculty.

This file must contain on the top:

Name:

Roll No.

Exp No.

Batch:

Date:

Question:

1. **Students have to explore different types of operating systems and differentiate them on the basis of pros and cons.**

Ans**: <u>Different types of OS are</u>**:

- Embedded Operating System:
  An Embedded Operating System is designed to perform a specific task for a particular device which is not a computer. For example, the software used in elevators is dedicated to the working of elevators only and nothing else. So, this can be an example of Embedded Operating System. The Embedded Operating System allows the access of device hardware to the software that is running on the top of the Operating System.

Pros	Cons
Since it is dedicated to a particular job, so it is fast.	Only one job can be performed.
Low cost.	It is difficult to upgrade or is nearly scalable.
These consume less memory and other resources.	

- Multiprogramming Operating System

  Multiprogramming is an extension to batch processing where the CPU is always kept busy. Each process needs two types of system time: CPU time and IO time.In a multiprogramming environment, when a process does its I/O, The CPU can start the execution of other processes. Therefore, multiprogramming improves the efficiency of the system.

Pros	Cons
Throughout the system, it increased as the CPU always had one program to execute	Multiprogramming system provide an environment which various systems resources are used efficiently, but they do not provide any user interaction with the computer system.
Response time can also be reduced	

- Multiprocessing Operating System

  In Multiprocessing, Parallel computing is achieved. There are more than one processor present in the system which can execute more than one process at the same time. This will increase the throughput of the system. In Multiprocessing, Parallel computing is achieved. More than one processor present in the system can execute more than one process simultaneously, which will increase the throughput of the system.

Pros	Cons
Due to the multiprocessing system, processing tasks can be distributed among several processors. This increases reliability as if one processor fails, the task can be given to another processor for completion.	Multiprocessing operating system is more complex and sophisticated as it takes care of multiple CPUs simultaneously.
As several processors increase, more work can be done in less.	

- Multitasking Operating System
  The multitasking operating system is a logical extension of a multiprogramming system that enables **multiple** programs simultaneously. It allows a user to perform more than one computer task at the same time.

Pros	Cons
This operating system is more suited to supporting multiple users simultaneously.	The multiple processors are busier at the same time to complete any task in a multitasking environment, so the CPU generates more heat.
The multitasking operating systems have well-defined memory management.	

<u>A comparative Study of Different OS : Windows v/s Linux v/s IOS</u>

Windows	Linux	IOS
It was developed and is owned by **Microsoft Incorporation**.	It was developed by **Linus Torvalds**.	It was developed by **Apple Incorporation**.
Kernel type is Hybrid with modules.	Its kernel type is Monolithic.	Its kernel type is Hybrid.
The native APIs are Win32 and NT API.	Its native APIs are LINUX/POSIX.	Its native APIs are Cocoa and BSD-POSIX.
Update management is Windows Update.	Its update management depends on the distribution	Its update management is Software Update.

File systems supported are NTFS, FAT, ISO 9660, UDF, HFS+, FATX and HFS.	File systems supported by Linux are ext2, ext3, ext4, btrfs, ReiserFS, FAT, ISO 9660, UDF and NFS.	File systems supported by iOS are HFS+ and APFS.
It is for workstation, personal computers, media center, tablets and embedded systems.	Its target system types are embedded systems, mobile devices, personal computers, servers, mainframe computers and supercomputers.	Its target system types are smartphone, music player and tablet computer.
It charges for original version.	The non-native APIs supported through its subsystems are Mono, Java, Win16 and Win32.	The non-native APIs are not supported through its subsystems.

2. **Take any 5 domains like gaming, finance, banking etc and suggest which OS will be best suitable for these domains.**

Ans: OS which are best suitable for 5 domains are:

- Hacking: KALI LINUX.
- Gaming: Win10
- Coding: GNU LINUX
- Business: Windows
- Banking: Windows/MAC

Commands

>>>Basic Commands

```
soham@Soham:~$ pwd

/home/soham

soham@Soham:~$ ls

soham@Soham:~$ ls -a

.    ..    .bash_history    .bash_logout    .bashrc    .landscape    .motd_shown    .profile
.sudo_as_admin_successful

soham@Soham:~$ history

    1  sudo apt get update

    2  sudo apt-get update

    3  sudo apt-get upgrade

    4  whoami

    5  passwd
    6  which python
    7  bash readme.sh
    8  bash readme.sh 4
    9  bash readme.sh 4 abc hkh Easy
   10  cat /etc/shells
   11  which bash
   12  ls
   13  touch helloScript.sh
   14  ls
   15  ls -al
   16  clear
   17  chmod +x helloScript.sh
   18  ls
   19  ls -al
   20  ./helloScript.sh
   21  ./helloScript.sh
   22  ls
   23  cd
   24  ls
   25  ls
   26  ls
   27  pwd
   28  ls
   29  pwd
   30  ls
   31  ls -a
   32  history
```

>>>Making file structure

```
soham@Soham:~$ mkdir A
soham@Soham:~$ ls
A
```

```
soham@Soham:~$ cd A
soham@Soham:~/A$ mkdir B
soham@Soham:~/A$ mkdir C
soham@Soham:~/A$ cd B
soham@Soham:~/A/B$ touch b_file.txt
soham@Soham:~/A/B$ gedit b_file.txt
  soham@Soham:~/A/B$ sudo gedit b_file.txt
sudo: gedit: command not found
soham@Soham:~/A/B$ sudo nano b_file.txt
soham@Soham:~/A/B$ cat b_file.txt
Hello there, this file is under B directory
soham@Soham:~/A/B$ cd ..
soham@Soham:~/A$ mkdir C
mkdir: cannot create directory ‘C’: File exists
soham@Soham:~/A$ ls
B  C
soham@Soham:~/A$ cd C
soham@Soham:~/A/C$ ls
soham@Soham:~/A/C$ touch c_file.txt
soham@Soham:~/A/C$ cat c_file.txt
soham@Soham:~/A/C$ nano c_file.txt
soham@Soham:~/A/C$ cat c_file.txt
this file is under C directory
soham@Soham:~/A/C$ ls -l c_file.txt
-rw-r--r-- 1 soham soham 31 Sep 12 01:57 c_file.txt
```

>>Permissions

```
soham@Soham:~/A/C$ chmod 644 c_file.txt
soham@Soham:~/A/C$ ls -l c_file.txt
-rw-r--r-- 1 soham soham 31 Sep 12 01:57 c_file.txt
soham@Soham:~/A/C$ chmod 764 c_file.txt
soham@Soham:~/A/C$ ls -l c_file.txt
```

```
-rwxrw-r-- 1 soham soham 31 Sep 12 01:57 c_file.txt
soham@Soham:~/A/C$ ls
c_file.txt
soham@Soham:~/A/C$ rm c_file.txt
soham@Soham:~/A/C$ ls
soham@Soham:~/A/C$ cd ..
soham@Soham:~/A$ ls
B  C
soham@Soham:~/A$ rmdir C

soham@Soham:~/A$ ls
B
soham@Soham:~/A$ cd B
soham@Soham:~/A/B$ rm b_file.txt
soham@Soham:~/A/B$ cd ..
soham@Soham:~/A$ rmdir B
soham@Soham:~/A$ cd ..
soham@Soham:~$ rmdir A
soham@Soham:~$ cd
soham@Soham:~$ ls
```

Outcomes: CO1: Understand basic structure of modern operating system.

Conclusion:

Students need to complete the activity and upload the document containing the cmds and their results.

References:

Books/ Journals/ Websites:

1. RichardBlumandChristineBresnahan, “LinuxCommandLine&ShellScripting”, IIndEditionedition, Wiley, 2012.

2. Guru99. (11 August 2020). File Permissions in Linux/Unix with Example. Retrieved from https://www.guru99.com/file-Permissions.html