Lab Report No:	03
Lab Report Name:	File Operation and permission
ID:	IT-17037

Objective:- In this lab we will explore how the operating system protects sensitive files and controls its access. File permission can be granted or denied depending on the use case, and we will review how these permissions can be modified by the user.

File Operation:-A file is an abstract data type. For defining a file properly, we need to consider the operations that can be performed on files. The operating system can provide system calls to create, write, read, reposition, delete, and truncate files. There are six basic file operations within an Operating system. These are:

- Creating a file: There are two steps necessary for creating a file. First, space in the file system must be found for the file. We discuss how to allocate space for the file. Second, an entry for the new file must be made in the directory.
- Writing a file: To write to a file, you make a system call specify about both the name of the file along with the information to be written to the file.
- Reading a file: To read from a file, you use a system call which specifies the name of the file and where within memory the next block of the file should be placed.
- Repositioning inside a file: The directory is then searched for the suitable entry, and the 'current-file-position' pointer is relocating to a given value. Relocating within a file need not require any actual I/O. This file operation is also termed as 'file seek.'

- Deleting a file: For deleting a file, you have to search the directory for the specific file. Deleting that file or directory release all file space so that other files can re-use that space.
- Truncating a file: The user may wish for erasing the contents of a file but keep the attributes same. Rather than deleting the file and then recreate it, this utility allows all attributes to remain unchanged — except the file length — and let the user add or edit the file content.

File Permission:- Linux is a multi-user operating system, so it has security to prevent people from accessing each other's confidential files. Each file and directory has three user based permission groups:

- **owner** – The Owner permissions apply only the owner of the file or directory, they will not impact the actions of other users.
- **group** – The Group permissions apply only to the group that has been assigned to the file or directory, they will not effect the actions of other users.
- **all users** – The All Users permissions apply to all other users on the system, this is the permission group that you want to watch the most.

Permission Types

Each file or directory has three basic permission types:

- **read** – The Read permission refers to a user's capability to read the contents of the file.
- **write** – The Write permissions refer to a user's capability to write or modify a file or directory.
- **execute** – The Execute permission affects a user's capability to execute a file or view the contents of a directory.

Implementation of file operation:-

Create a file

```
rafatul@rafatul-HP-Notebook:~/Downloads$ touch new.txt
rafatul@rafatul-HP-Notebook:~/Downloads$ ls
new.txt
```

Write a file

```
rafatul@rafatul-HP-Notebook:~$ write
usage: write user [tty]
rafatul@rafatul-HP-Notebook:~$ write hello world
write: hello is not logged in on world
rafatul@rafatul-HP-Notebook:~$
```

Read a file

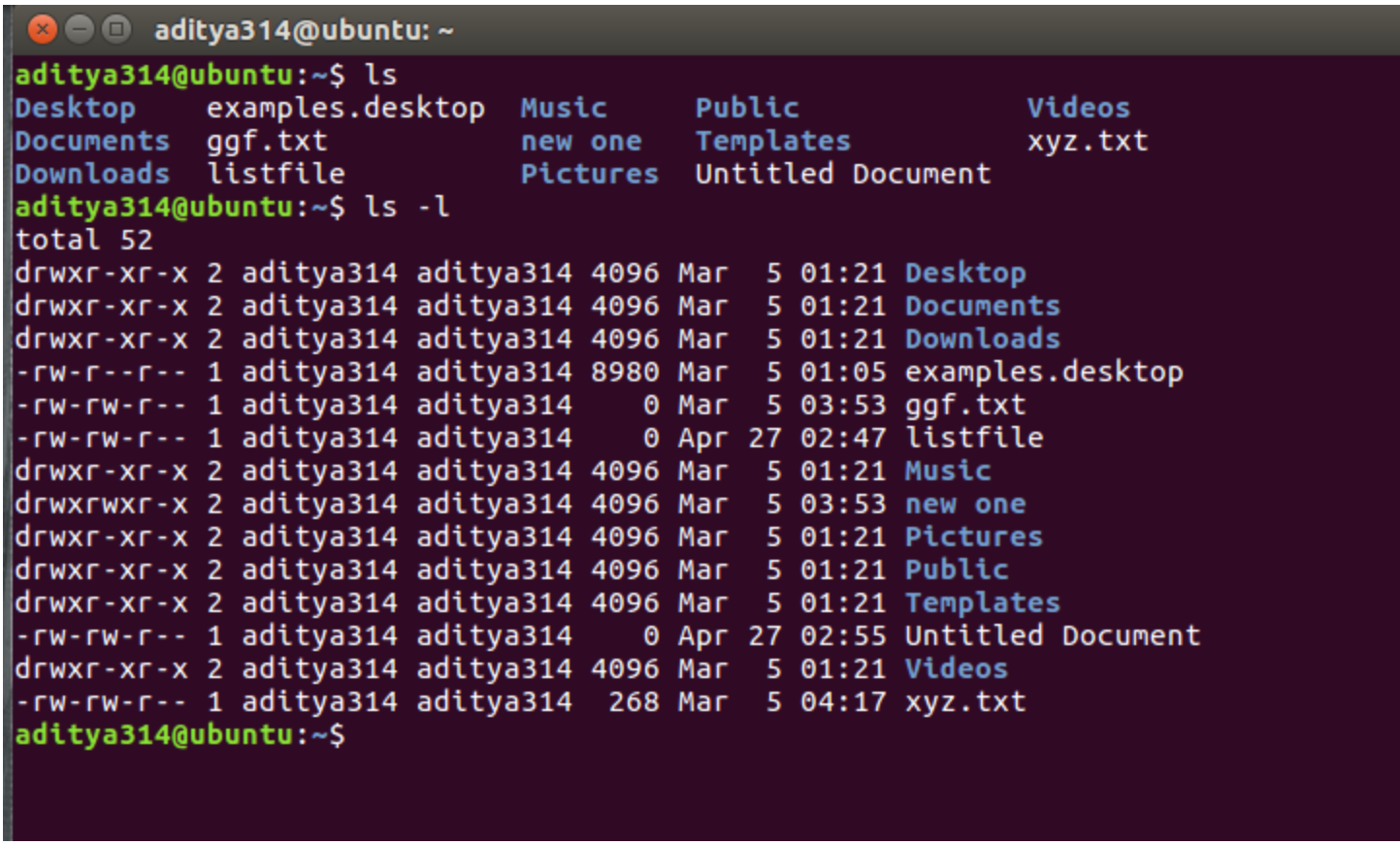
```
rafatul@rafatul-HP-Notebook:~$ echo "what is your name...?"
what is your name...?
rafatul@rafatul-HP-Notebook:~$
```

Repositioning a file

```
rafatul@rafatul-HP-Notebook:~/Downloads$ mv new.txt newer.txt
rafatul@rafatul-HP-Notebook:~/Downloads$ ls
Music  newer.txt
```

Implementation of file permission:-Viewing the Permissions

You can view the permissions by checking the file or directory permissions in your favorite GUI File Manager (which I will not cover here) or by reviewing the output of the ***"ls -l"*** command while in the terminal and while working in the directory which contains the file or folder.



The permission in the command line is displayed as: ***_rwxrwxrwx 1 owner:group***

1. User rights/Permissions
    1. The first character that I marked with an underscore is the special permission flag that can vary.
    2. The following set of three characters (rwx) is for the owner permissions.
    3. The second set of three characters (rwx) is for the Group permissions.
    4. The third set of three characters (rwx) is for the All Users permissions.
2. Following that grouping since the integer/number displays the number of hard links to the file.
3. The last piece is the Owner and Group assignment formatted as Owner:Group.

Modifying the Permissions

When in the command line, the permissions are edited by using the command ***chmod***. You can assign the permissions explicitly or by using a binary reference as described below.

.

```
aditya314@ubuntu:~$ chmod ugo-rwx xyz.txt
aditya314@ubuntu:~$ ls -l
total 52
drwxr-xr-x 2 aditya314 aditya314 4096 Mar  5 01:21 Desktop
drwxr-xr-x 2 aditya314 aditya314 4096 Mar  5 01:21 Documents
drwxr-xr-x 2 aditya314 aditya314 4096 Mar  5 01:21 Downloads
-rw-r--r-- 1 aditya314 aditya314 8980 Mar  5 01:05 examples.desktop
-rw-rw-r-- 1 aditya314 aditya314    0 Mar  5 03:53 ggf.txt
-rw-rw-r-- 1 aditya314 aditya314    0 Apr 27 02:47 listfile
drwxr-xr-x 2 aditya314 aditya314 4096 Mar  5 01:21 Music
drwxrwxr-x 2 aditya314 aditya314 4096 Mar  5 03:53 new one
drwxr-xr-x 2 aditya314 aditya314 4096 Mar  5 01:21 Pictures
drwxr-xr-x 2 aditya314 aditya314 4096 Mar  5 01:21 Public
drwxr-xr-x 2 aditya314 aditya314 4096 Mar  5 01:21 Templates
-rw-rw-r-- 1 aditya314 aditya314    0 Apr 27 02:55 Untitled Document
drwxr-xr-x 2 aditya314 aditya314 4096 Mar  5 01:21 Videos
---------- 1 aditya314 aditya314  268 Mar  5 04:17 xyz.txt
aditya314@ubuntu:~$
```

Explicitly Defining Permissions

To explicity define permissions you will need to reference the Permission Group and Permission Types.

The Permission Groups used are:

u – Owner
g – Group
o – Others
a – All users

The potential Assignment Operators are + (plus) and – (minus); these are used to tell the system whether to add or remove the specific permissions.

The Permission Types that are used are:

- **r** – Read
- **w** – Write
- **x** – Execute

So for an example, lets say I have a file named file1 that currently has the permissions set to **_rw_rw_rw,** which means that the owner, group and all users have read and write permission. Now we want to remove the read and write permissions from the all users group.

To make this modification you would invoke the command: ***chmod a-rw file1***
To add the permissions above you would invoke the command: ***chmod a+rw file1***

As you can see, if you want to grant those permissions you would change the minus character to a plus to add those permissions.

Conclusion:-With a basic understanding of how file permissions work in Linux, you are better prepared to secure files from accidental or malicious harm. You can also keep an eye out for errors that are caused by restrictive file permissions, such as an application being unable to write to its log (caused by having no write permission for the user that owns the process), or a web server that is unable to serve an html file (caused by having no read permission, or the directory doesn't have execute permission).