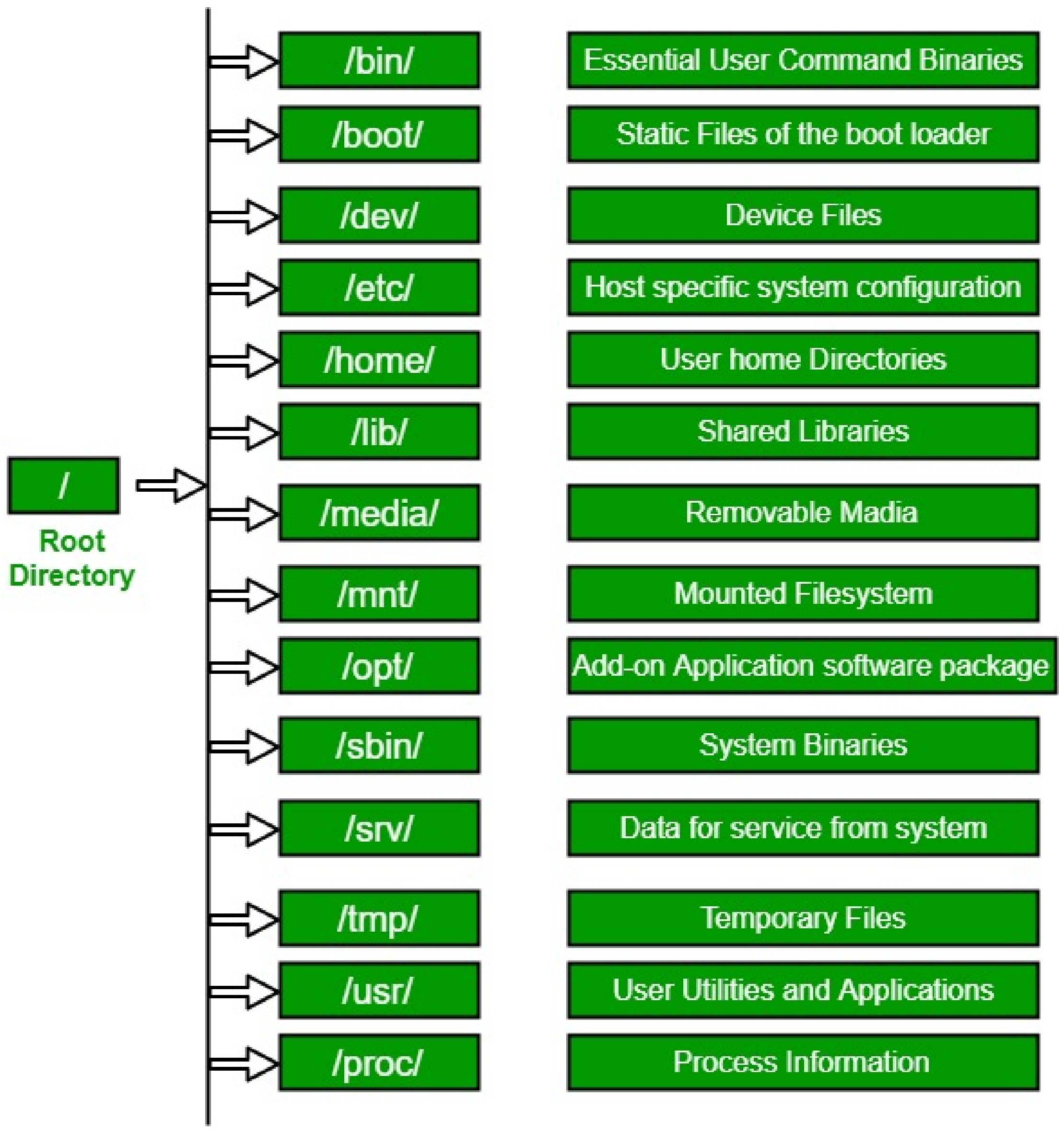
Experiment -3

Aim :- File system hierarchy in a common Linux distribution, file and device permissions, study of system configuration files in /etc, familiarizing log files for system events, user activity, network events.

File System Hierarchy in Linux :-

The Linux File Hierarchy Structure or the File system Hierarchy Standard (FHS) defines the directory structure and directory contents in Unix-like operating systems. It is maintained by the Linux Foundation.

- In the FHS, all files and directories appear under the root directory /, even if they are stored on different physical or virtual devices.
- Some of these directories only exist on a particular system if certain subsystems, such as the X Window System, are installed.
- Most of these directories exist in all UNIX operating systems and are generally used in much the same way; however, the descriptions here are those used specifically for the FHS and are not considered authoritative for platforms other than Linux.



1. / (Root): Primary hierarchy root and root directory of the entire file system hierarchy.

- Every single file and directory starts from the root directory.
- The only root user has the right to write under this directory.
- /root is the root user's home directory, which is not the same as /.

2. /bin : Essential command binaries that need to be available in single-user mode for all users, e.g., cat, ls, cp. .

- Contains binary executables.
- Common linux commands you need to use in single-user modes are located under this directory.
- Commands used by all the users of the system are located here e.g. ps, ls, ping, grep, cp.

3. /boot : Boot loader files, e.g., kernels, initrd.

- Kernel initrd, vmlinux, grub files are located under /boot
- Example: initrd.img-2.6.32-24-generic, vmlinuz-2.6.32-24-generic.

4. /dev : Essential device files, e.g., /dev/null.

- These include terminal devices, usb, or any device attached to the system.

  Example: /dev/tty1, /dev/usbmon0

5. /etc : Host-specific system-wide configuration files.

- Contains configuration files required by all programs.
- This also contains startup and shutdown shell scripts used to start/stop individual programs.

  Example: /etc/resolv.conf, /etc/logrotate.conf.

6. /home : Users' home directories, containing saved files, personal settings, etc.

- Home directories for all users to store their personal files.

  example: /home/kishlay, /home/kv

7. /lib : Libraries essential for the binaries in /bin/ and /sbin/.

- Library filenames are either ld* or lib*.so.*

Example: ld-2.11.1.so, libncurses.so.5.7

8. /mnt : Temporarily mounted filesystems.

- Temporary mount directory where sysadmins can mount filesystems.

9. /opt : Optional application software packages.

- Contains add-on applications from individual vendors.
- Add-on applications should be installed under either /opt/ or /opt/ sub-directory.

10. /tmp : Temporary files. Often not preserved between system reboots, and may be severely size restricted.

- Directory that contains temporary files created by system and users.
- Files under this directory are deleted when system is rebooted.

11. /usr : Secondary hierarchy for read-only user data; contains the majority of (multi-)user utilities and applications.

- Contains binaries, libraries, documentation, and source-code for second level programs.
- /usr/bin contains binary files for user programs. If you can't find a user binary under /bin, look under /usr/bin. For example: at, awk, cc, less, scp.
- /usr/sbin contains binary files for system administrators. If you can't find a system binary under /sbin, look under /usr/sbin. For example: atd, cron, sshd, useradd, userdel.
- /usr/lib contains libraries for /usr/bin and /usr/sbin.

- /usr/local contains users programs that you install from source. For example, when you install apache from source, it goes under /usr/local/apache2.
- /usr/src holds the Linux kernel sources, header-files and documentation.

12. /proc : Virtual filesystem providing process and kernel information as files. In Linux, corresponds to a procfs mount. Generally automatically generated and populated by the system, on the fly.

- Contains information about system process.
- This is a pseudo filesystem contains information about running process. For example: /proc/{pid} directory contains information about the process with that particular pid.
- This is a virtual filesystem with text information about system resources. For example: /proc/uptime

File And Device Permissions :-

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

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 Linux group permissions access 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 he belongs 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.

File Permissions :-

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

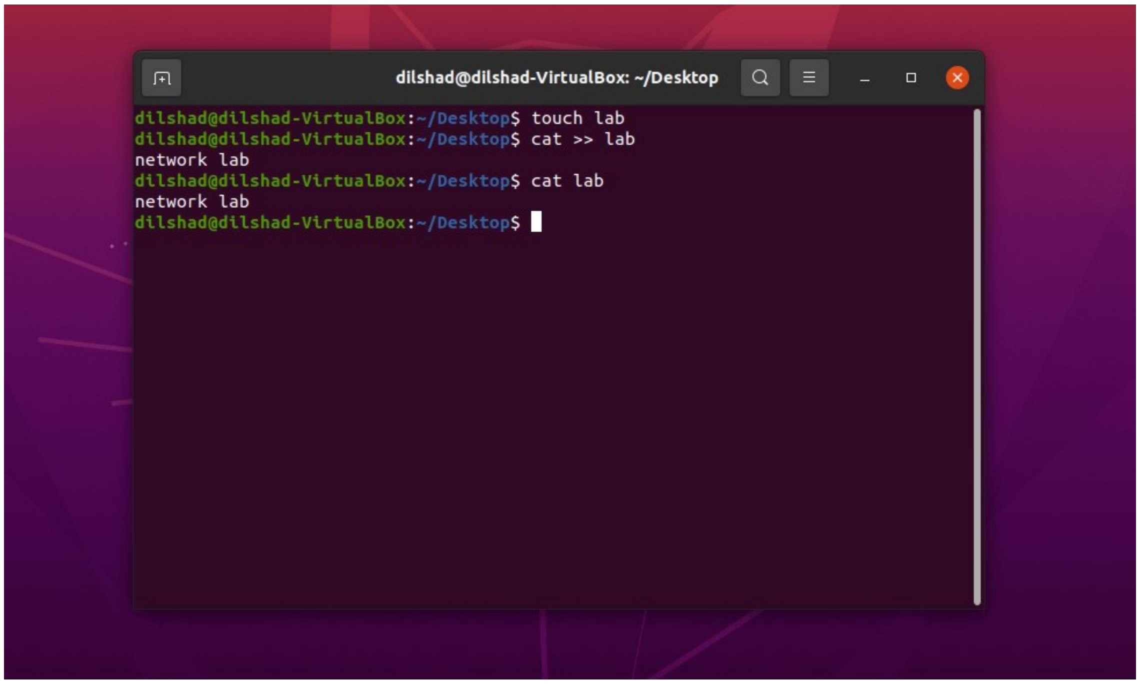
Read: This permission give you the authority to open and read a file. Read permission on a directory gives you the ability to lists its content.

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.

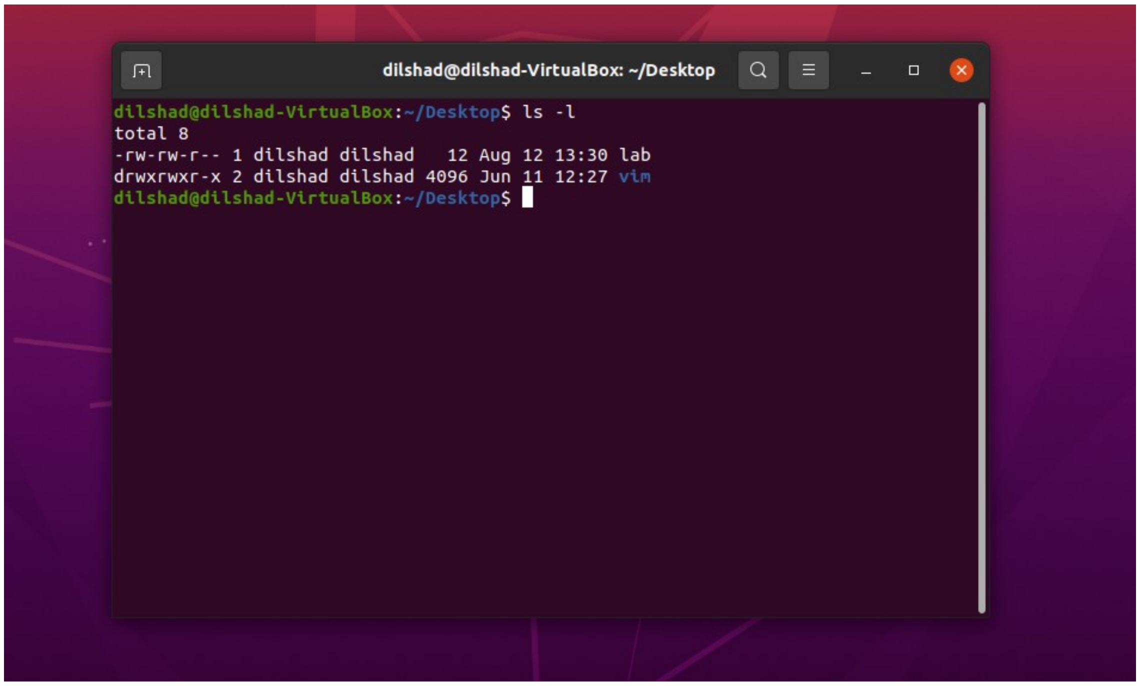
Example :-

Create a new file and add some text in it.



```
dilshad@dilshad-VirtualBox:~/Desktop$ touch lab
dilshad@dilshad-VirtualBox:~/Desktop$ cat >> lab
network lab
dilshad@dilshad-VirtualBox:~/Desktop$ cat lab
network lab
dilshad@dilshad-VirtualBox:~/Desktop$
```

Use the command ***ls –l*** to show the current permission details of the file.



```
dilshad@dilshad-VirtualBox:~/Desktop$ ls -l
total 8
-rw-rw-r-- 1 dilshad dilshad   12 Aug 12 13:30 lab
drwxrwxr-x 2 dilshad dilshad 4096 Jun 11 12:27 vim
dilshad@dilshad-VirtualBox:~/Desktop$
```

⇒ It shows a minus(-) In front of permission of lab and *d* for the vim. This is to show that minus(-) determines it as a file and *d* determines it as a directory.

⇒ In the above example the file lab have the permission with 3 bit representation for the 3 types of owners.

⇒ First 3 bit for user,then next 3 bit for group and last 3 bit for others/world.

⇒ Here user have read and write permission on lab file,group also has read and write permission on file and others can have only read permission.

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	Read	r--
5	Read + Execute	r-x
6	Read +Write	rw-
7	Read + Write +Execute	Rwx

Symbolic Mode

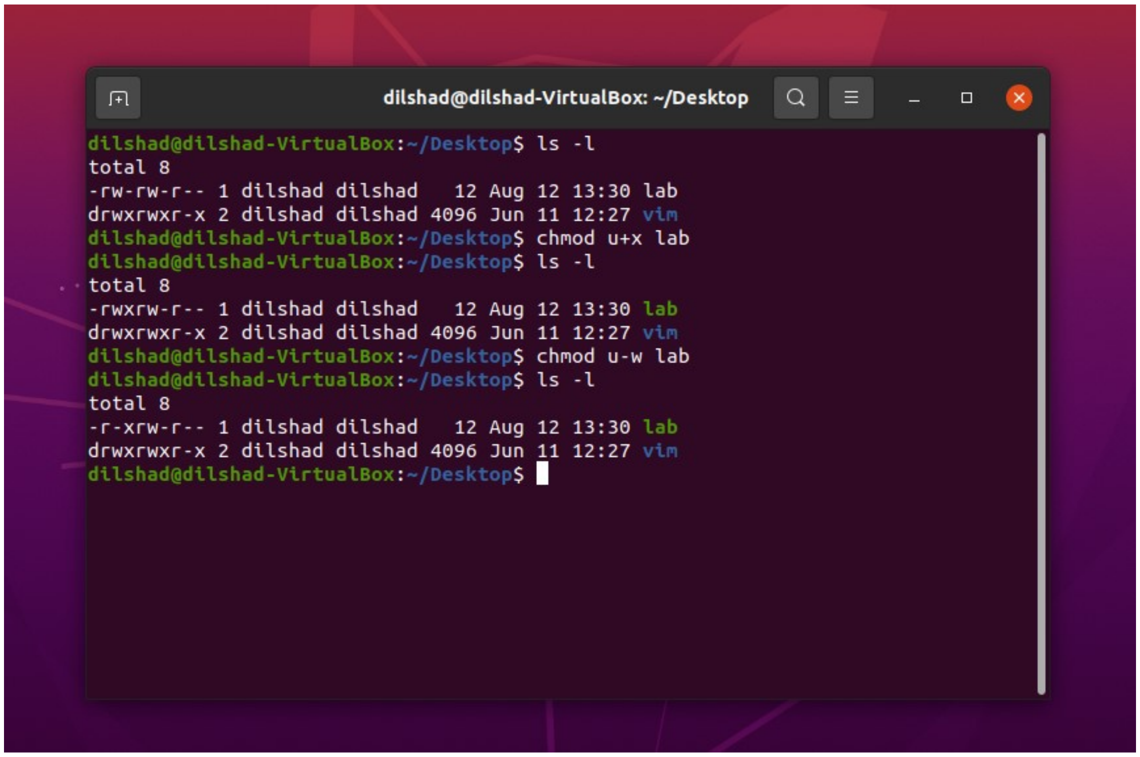
In the Absolute mode, you change permissions for all 3 owners. In the symbolic mode, you can modify permissions of a specific owner. It makes use of mathematical symbols to modify the Unix file permissions.

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

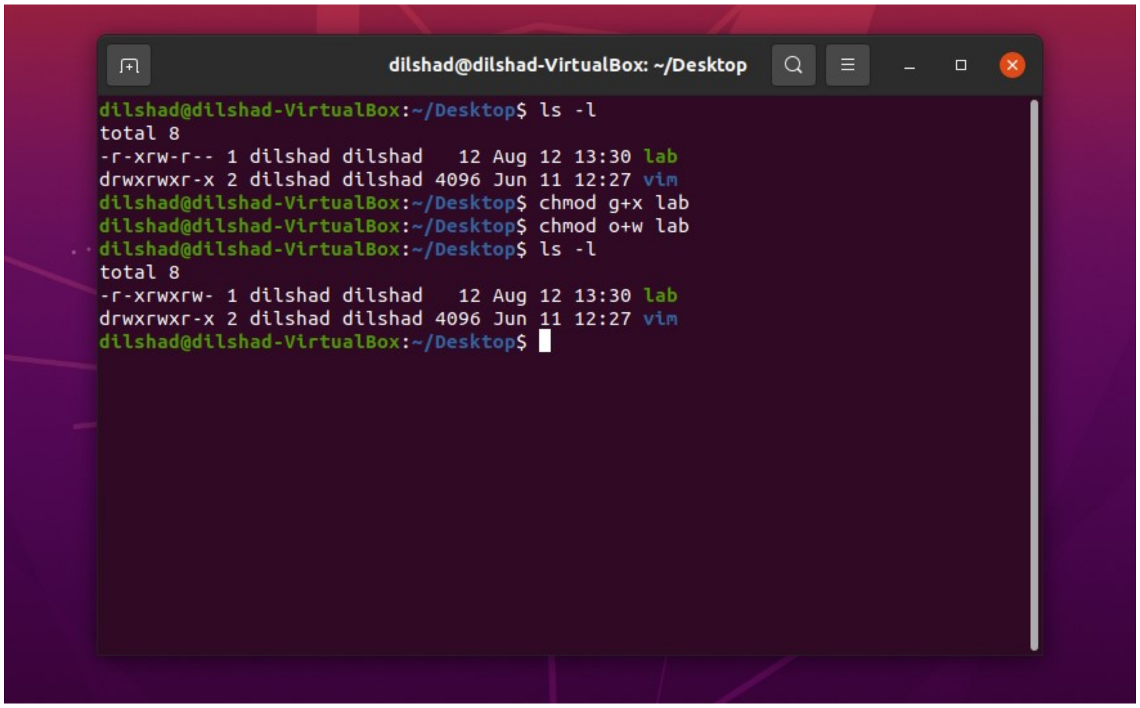
Example 1 (symbolic mode):- Chmod is the command used to change permission mode of a file.



```
dilshad@dilshad-VirtualBox:~/Desktop$ ls -l
total 8
-rw-rw-r-- 1 dilshad dilshad   12 Aug 12 13:30 lab
drwxrwxr-x 2 dilshad dilshad 4096 Jun 11 12:27 vim
dilshad@dilshad-VirtualBox:~/Desktop$ chmod u+x lab
dilshad@dilshad-VirtualBox:~/Desktop$ ls -l
total 8
-rwxrw-r-- 1 dilshad dilshad   12 Aug 12 13:30 lab
drwxrwxr-x 2 dilshad dilshad 4096 Jun 11 12:27 vim
dilshad@dilshad-VirtualBox:~/Desktop$ chmod u-w lab
dilshad@dilshad-VirtualBox:~/Desktop$ ls -l
total 8
-r-xrw-r-- 1 dilshad dilshad   12 Aug 12 13:30 lab
drwxrwxr-x 2 dilshad dilshad 4096 Jun 11 12:27 vim
dilshad@dilshad-VirtualBox:~/Desktop$
```

In symbolic mode + uses to add a permission and – uses for removing a permission.

In the example we are adding execute permission to the user.And removing write permission from user.

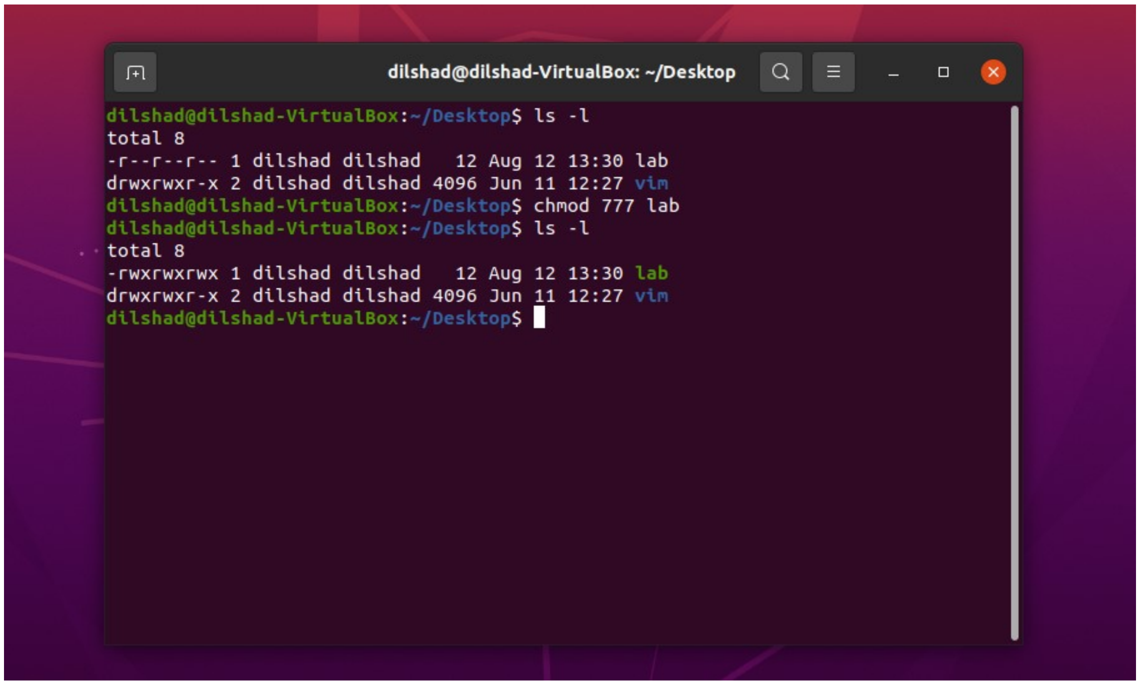


```
dilshad@dilshad-VirtualBox:~/Desktop$ ls -l
total 8
-r-xrw-r-- 1 dilshad dilshad   12 Aug 12 13:30 lab
drwxrwxr-x 2 dilshad dilshad 4096 Jun 11 12:27 vim
dilshad@dilshad-VirtualBox:~/Desktop$ chmod g+x lab
dilshad@dilshad-VirtualBox:~/Desktop$ chmod o+w lab
dilshad@dilshad-VirtualBox:~/Desktop$ ls -l
total 8
-r-xrwxrw- 1 dilshad dilshad   12 Aug 12 13:30 lab
drwxrwxr-x 2 dilshad dilshad 4096 Jun 11 12:27 vim
dilshad@dilshad-VirtualBox:~/Desktop$ 
```

In this we are giving execute permission to the group and write permission to the others.

Example 2 :-

With the absolute(numeric) mode we are giving all owners the permission to read , write and execute the file



```
dilshad@dilshad-VirtualBox:~/Desktop$ ls -l
total 8
-r--r--r-- 1 dilshad dilshad   12 Aug 12 13:30 lab
drwxrwxr-x 2 dilshad dilshad 4096 Jun 11 12:27 vim
dilshad@dilshad-VirtualBox:~/Desktop$ chmod 777 lab
dilshad@dilshad-VirtualBox:~/Desktop$ ls -l
total 8
-rwxrwxrwx 1 dilshad dilshad   12 Aug 12 13:30 lab
drwxrwxr-x 2 dilshad dilshad 4096 Jun 11 12:27 vim
dilshad@dilshad-VirtualBox:~/Desktop$
```

System configuration files in /etc :-

The /etc hierarchy contains configuration files. A "configuration file" is a local file used to control the operation of a program; it must be static and cannot be an executable binary.

It is recommended that files be stored in subdirectories of /etc rather than directly in /etc.

/etc/opt

Host-specific configuration files for add-on application software packages must be installed within the directory /etc/opt/<subdir>, where <subdir> is the name of the subtree in /opt where the static data from that package is stored.

/etc/X11

/etc/X11 is the location for all X11 host-specific configuration. This directory is necessary to allow local control if */usr* is mounted read only.

/etc/sgml

Generic configuration files defining high-level parameters of the SGML systems are installed here. Files with names *.conf indicate generic configuration files. File with names *.cat are the DTD-specific centralized catalogs, containing references to all other catalogs needed to use the given DTD. The super catalog file catalog references all the centralized catalogs.

/etc/xml

Generic configuration files defining high-level parameters of the XML systems are installed here. Files with names *.conf indicate generic configuration files. The super catalog file catalog references all the centralized catalogs.

Log Files :-

Log files are the records that Linux stores for administrators to keep track and monitor important events about the server, kernel, services, and applications running on it.

Linux provides a centralized repository of log files that can be located under the **/var/log** directory.

/var/log/messages

- This log file contains generic system activity logs.
- It is mainly used to store informational and non-critical system messages.

/var/log/auth.log

- All authentication related events in Debian and Ubuntu server are logged here.
- If you're looking for anything involving the user authorization mechanism, you can find it in this log file.

/var/log/secure

- It is mainly used to track the usage of authorization systems.
- It stores all security related messages including authentication failures.
- It also tracks sudo logins, SSH logins and other errors logged by system security services daemon.

/var/log/faillog

This file contains information on failed login attempts.

/var/log/mail.log

All mail server related logs are stored here.

/var/log/mysql.log

- As the name suggests, this is the MySQL log file.
- All debug, failure and success messages related to the [mysqld] and [mysqld_safe] daemon are logged to this file.