1. Introduction and terminology

Rembember

A user is anyone who has Unix account on the system. Unix recognizes a user by a number called user id.

A super user or root:

- Has the maximum set of privileges in the system
- Also know as system administrator
- Can change the system
- Must have a lot of experience and training

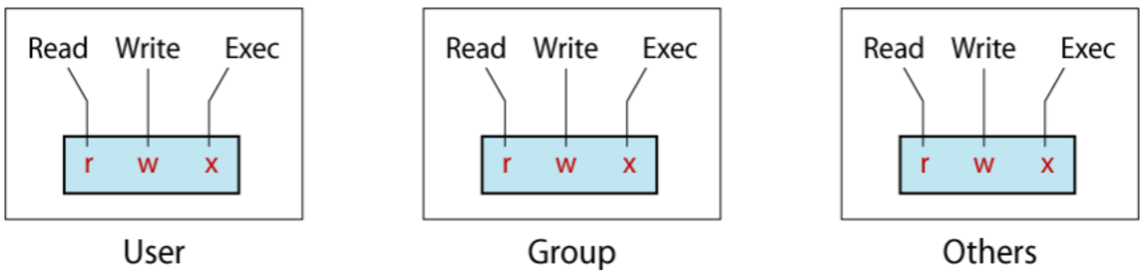
Users can be organized into groups. One or more users can belong to multiple groups.

Access Permission Code

The protection on a file is referred to as its file modes. Linux supports three types of access permissions:

- **r** read
- **w** write
- **x** execute
- **-** Permission denied

Linux assigns different permission to owner, group and other users.



Access types

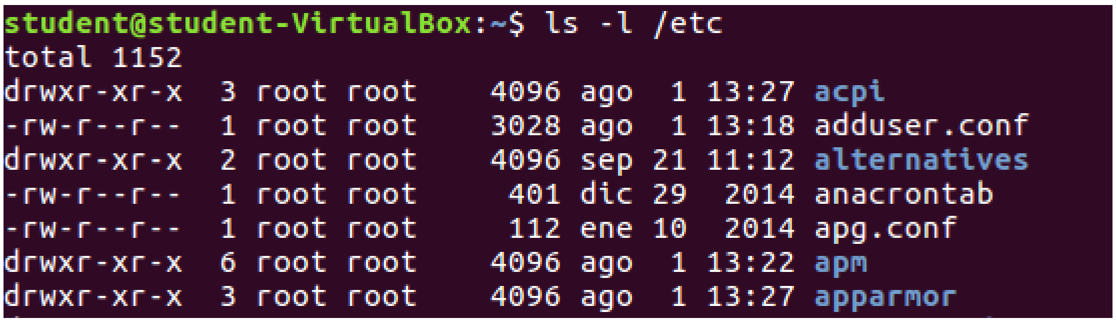
The meaning of each permission will be different depending on whether they are assigned to a file or directory.

Permission	File	Directory
read	User can look at the contents of the file	User can list the files in the directory
write	User can modify the contents of the file	User can create new files and remove existing files in the directory
execute	User can use the filename as a UNIX command	User can change into the directory, but cannot list the files unless (s)he has read permission. User can read files if (s)he has read permission on them.

Checking Permissions

To check the permissions of an existing file or an existing directory, use the command: ls –l

Example: ls –l /etc

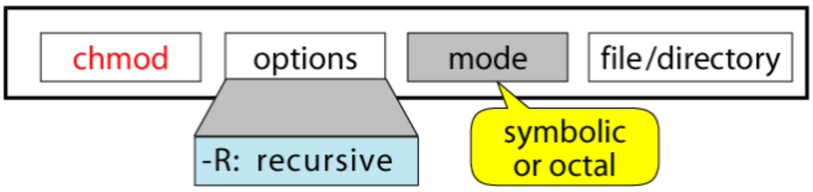


```
student@student-VirtualBox:~$ ls -l /etc
total 1152
drwxr-xr-x  3 root root    4096 ago  1 13:27 acpi
-rw-r--r--  1 root root    3028 ago  1 13:18 adduser.conf
drwxr-xr-x  2 root root    4096 sep 21 11:12 alternatives
-rw-r--r--  1 root root     401 dic 29  2014 anacrontab
-rw-r--r--  1 root root     112 ene 10  2014 apg.conf
drwxr-xr-x  6 root root    4096 ago  1 13:22 apm
drwxr-xr-x  3 root root    4096 ago  1 13:27 apparmor
```

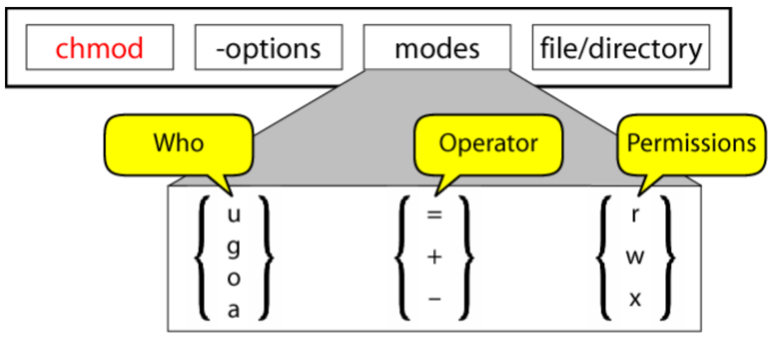
2. Changing permissions: chmod

File and directory permissions can only be modified by their owners, or by the superuser (root), by using the chmod system utility.

The **chmod** command accepts options in two forms: symbolic and octal.



Symbolic mode



Permissions may be specified symbolically, using the symbols u (user), g (group), o (other), a (all), r (read), w (write), x (execute), + (add permission), - (take away permission) and = (assign permission). For example, the command:

```
chmod ug=rw,o-rw,a-x *.txt
```

It sets the permissions on all files ending in *.txt to rw-rw---- (i.e. the owner and users in the file's group can read and write to the file, while the general public do not have any sort of access).

Another example:

To change the permissions on the file “sort.c” using symbolic mode, so that:

a) Everyone may read and execute it
b) Only the owner and group may write to it.
c) So, we want this using symbolic mode: rwx rwx r-x

```
chmod ug=rwx,o=rx sort.c
```

Now, if we want to remove “w” permission from all the users:

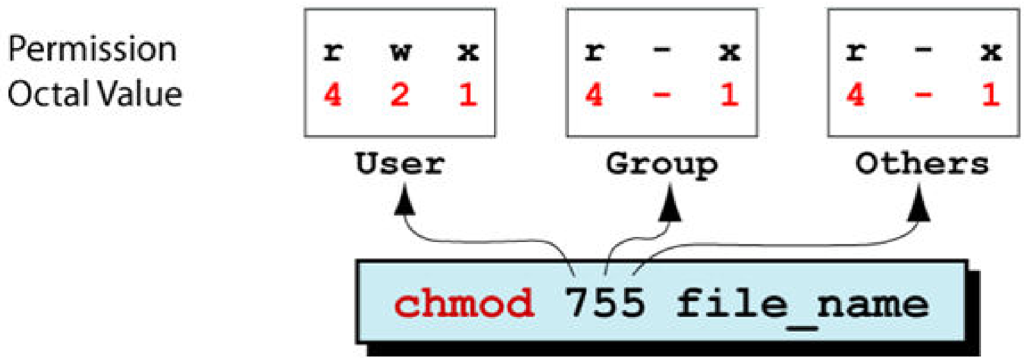
```
chmod a-w sort.c
```

Octal mode

Permissions may also be specified as a sequence of 3 octal digits. Each octal digit represents the access permissions for the user/owner, group and others respectively. The mappings of permissions onto their corresponding octal digits is as follows:

---	**0**
--x	**1**
-w-	**2**
-wx	**3**
r--	**4**
r-x	**5**
rw-	**6**
rwx	**7**

Example 1:



Example 2: Only the owner can read and write to the file

```
chmod 600 private.txt
```

chmod also supports a -R option which can be used to recursively modify file permissions, e.g.

```
chmod -R go+r play
```

The command above will grant group and other read rights to the directory play and all of the files and directories within play.

3. Changing owner user and group

Chown command changes the user and/or group ownership of for given file. This command can only be executed by root.

The syntax is:

```
chown owner-user file
chown owner-user:owner-group file
chown owner-user:owner-group directory
```

Examples

First, list permissions for demo.txt, enter:

```
ls -l demo.txt
```

Sample outputs:

```
-rw-r--r-- 1 root root 0 Aug 31 05:48 demo.txt
```

In this example change file ownership to vivek user and list the permissions, run:
chown vivek demo.txt

```
ls -l demo.txt
```

Sample outputs:

```
-rw-r--r-- 1 vivek root 0 Aug 31 05:48 demo.txt
```

In this next example, the owner is set to vivek followed by a colon and a group onwership is also set to vivek group, run:

```
chown vivek:vivek demo.txt
ls -l demo.txt
```

Sample outputs:

```
-rw-r--r-- 1 vivek vivek 0 Aug 31 05:48 demo.txt
```

In this example, change only the group of file. To do so, the colon and following GROUP-name ftp are given, but the owner is omitted, only the group of the files is changed:

```
chown :ftp demo.txt
ls -l demo.txt
```

Sample outputs:

```
-rw-r--r-- 1 vivek ftp 0 Aug 31 05:48 demo.txt
```

chgrp (change group) can also be used to change the group that a file or directory belongs to.

```
chgrp vivek demo.txt
```

Notice that both **chown** and **chgrp** can also supports a **-R** option to to recursively change user or group ownership.