Understanding Linux File Permissions

Ferdowsi University of Mashhad Linux Users Group

t.me/FUMLUG

$ chmod

Understanding Linux File Permissions :

Although there are already a lot of good security features built into Linux-based systems, one very important potential vulnerability can exist when local access is granted - - that is file permission based issues resulting from a user not assigning the correct permissions to files and directories. So based upon the need for proper permissions, I will go over the ways to assign permissions and show you some examples where modification may be necessary.

Note : in Linux world everything is “File” or “Process”.

$ chmod

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.

$ chmod

When applying permissions to directories on Linux, the permission bits have different meanings than on regular files.

- The read bit allows the affected user to list the files within the directory
- The write bit allows the affected user to create, rename, or delete files within the directory, and modify the directory's attributes
- The execute bit allows the affected user to enter the directory, and access files and directories inside

$ chmod

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 other users** - The All other Users permissions apply to all other users on the system, this is the permission group that you want to watch the most.

Viewing the Permissions :

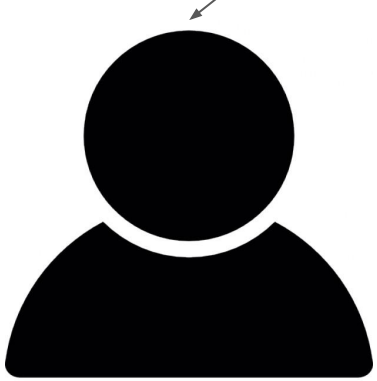
```
$ ls -l
```

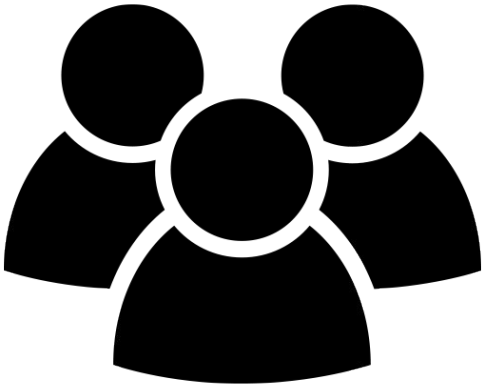
Viewing the Permissions Like Ninjas :

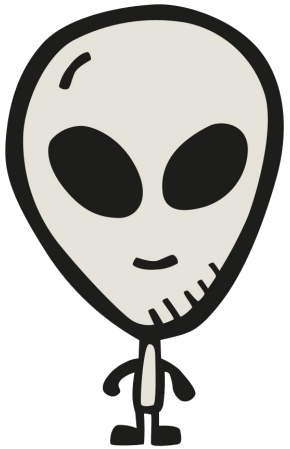
```
$ ls -ltrh
```

$ chmod

```
$ -      rwx      rwx      rwx
```







User(Owner)

Group

Others

$ chmod

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 hardlinks to the file.
3. The last piece is the Owner and Group assignment formatted as Owner:Group.

$ chmod

Changing permissions :

It is possible to change the permissions on files & directories using the chmod command. There are to ways to tell this command what you want to do:

1. using octal codes
2. using short coeds

When using octal codes, you have to to create an octal string to tell chmod what you want to do. This way, 0 means no access, means execute, 2 means write and 4 means read. So if you want to give read+execute, you have to give 4+1 which is 5. This table shows every possible combination:

$ chmod

```
$ ls -ltrh myfile
-rwxr-x--x 1 alizeyn alizeyn 0 Feb  8 21:01 myfile

$ chmod u-x myfile

$ ls -ltrh myfile
-rw-r-x--x 1 alizeyn alizeyn 0 Feb  8 21:01 myfile

$ chmod +x myfile

$ chmod uo+xr myfile

$ ls -ltrh myfile
-rwxr-xr-x 1 alizeyn alizeyn 0 Feb  8 21:01 myfile
```

$ chmod

r	w	x
4	2	1

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

$ chmod

```
$ ls -ltrh myfile
-rw-rw-r-- 1 alizeyn alizeyn 0 Feb  8 21:01 myfile
$ chmod 751 myfile
$ ls -ltrh myfile
-rwxr-x--x 1 alizeyn alizeyn 0 Feb  8 21:01 myfile
```

$ chown

Changing Owner or group :

If you need to change the ownership or group belonging of a file or directory, use the `chown` command:

Example :

```
$ ls -ltrh newfile
-rw-rw-r-- 1 alizeyn alizeyn 0 Feb  8 21:38 newfile
$ chown root:root newfile
chown: changing ownership of ‘newfile’: Operation not permitted
$ sudo chown root:root newfile
[sudo] password for alizeyn:
$ ls -ltrh newfile
-rw-rw-r-- 1 root root 0 Feb  8 21:38 newfile
```

A common switch is -R to do the chown recursively and the general style is chown `newuser:newgroup` file.

Presented by Ali Zeynali in GNU/Linux Workshop (Ferdowsi University of Mashhad)

ali.zzeynali@gmail.com
License: CC-BY 3.0

$ command

Title :

A long description about command or anything you trying to say, this slide is a template if you want to use this template for your presentation, do it freely :)

Example :

```
$ a line of shell command or bash script
```