

GETTING STARTED
WITH
FILE PERMISSIONS

➤ **PERMISSIONS:**

- Permission is an important component of LINUX that provides a **secure method for storing files**.
- Linux is a multi-user **O/S**, so it has security to prevent people from accessing each other's confidential files.
- Each file and directory have three user-based permissions:
 - Owner/User Permissions
 - Group Permissions
 - Other Permissions

Access level	Symbol
Read	r
Write	w
Execute	x

Each identity has a symbol:

Identity	Symbol
User	u
Group	g
Others	o

There are also operators to manipulate the permissions:

Task	Operator
Grant a level of access	+
Remove a level of access	-
Set a level of access	=

FILE ACCESS MODES:

- READ** : Grants the capability to read, i.e., view the contents of the file.
- WRITE** : Grants the capability to modify/remove the content of the file.
- EXECUTE** : User with execute permissions can run a file as a program.

→ To view Linux file / Directory permissions:

```
#ls -l  
  
-rwxrw-r-- 1 root root 2596 Oct 10 08:52 aws  
-rw-r--r-- 1 root root 32 Oct 10 08:25 devops  
drwxr-xr-x 2 root root 6 Oct 5 08:13 cloud
```

SYMBOLIC MODE:

rw-rw-r--

rw- : User Permissions

rw- : Group permissions

r-- : Others permissions

OCTAL MODE:

Owner: rwx $= 4+2+1 = 7$

Group: rw- $= 4+2+0 = 6$

Others: r-- $= 4+0+0 = 4$

UMASK:

- **UMASK** in Linux is known as **User Mask** or it is also called **User File creation MASK**.
- It is a command that determines the settings of a mask that controls which file permissions are set for files and directories when they are created.

→ To verify default umask value:

```
#umask
```

→ To change umask value:

```
#umask 134
```

```
#umask
```

CHMOD:

- It is used to modify file and directory permissions.
- chmod command also known as “Change Mode”.

SYNTAX: `#chmod [options] <permissions> file/directory`

SYMBOLIC METHOD:

→ Grant the read and write permissions to the group for cloud:

```
#chmod g+rw cloud  
#ls -ld cloud
```

→ Grant the only execute permissions to the others for aws file:

```
#chmod o+x aws  
#ls -l aws
```

→ Remove the read permissions from others for aws file:

```
#chmod o-r aws
```

→ To override existing permissions for aws file:

```
#chmod ugo=r aws
```

→ To set write permissions for all:

```
#chmod ugo+w aws
```

→ To change at a time all permissions:

```
#chmod u+x,g-w,o+wx aws
```

OCTAL METHOD:

→ Grant the read,write for user, readonly for group and execute for others:

```
#chmod 641 aws
```

→ Grant the read and execute for user only:

```
#chmod 500 aws
```

→ Grant the read only for the user for aws file:

#chmod 400 aws

→ Set read,write and execute for cloud folder in group level:

#chmod 070 cloud

→ Setting reference instead of mode values

#chmod -r aws devops

→ Setting full permissions of a folder:

#chmod 777 cloud

→ Only execute permissions for others of aws file:

#chmod 1 aws

→ To set recursive permissions for a folder cloud:

#chmod -R 751 cloud

#ls -ld cloud

#ls -l cloud

CHGRP:

- It is used to change group ownership.

SYNTAX: **#chgrp [options] <permissions> file/directory**

→ To change group name:

#chgrp family aws

#ls -l aws

CHOWN:

- To change file owner and group name.

SYNTAX: **#chown [options] <permissions> file/directory**

#chown raju aws

→ To change ownername and groupname:

#chmown raju:family aws

❖ GROUP COLLABORATION:

- Group Collaboration is an essential part of any system administrator who deals with users. Knowing file and directory permissions will allow you to let users work together.

SUID (SETUSERID):

- It is used to allow multi user access.
- Flag id value is: 4

→ Create a script with name script.sh:

```
#touch script.sh  
#ls -l script.sh
```

→ To setting suid for script.sh file:

```
#chmod u+s script.sh (or) #chmod 4644 script.sh  
#ls -l script.sh
```

NOTE: The **s** where **x** would usually indicate execute permissions for user.

→ To disable suid for script.sh file:

```
#chmod u-s script.sh (or) #chmod 0644 script.sh  
#ls -l script.sh
```

SGID (SETGROUPID):

- Used to allow multi group access.
- Flag id value is: 2

→ To setting up sgid for script.sh file:

```
#chmod g+s script.sh (or) #chmod 2644 script.sh  
#ls -l script.sh
```

NOTE: The **s** where **x** would usually indicate execute permissions for group.

→ To disable sgid for script.sh file:

```
#chmod g-s script.sh (or) #chmod 0644 script.sh  
#ls -l script.sh
```

STICKY BIT:

- It is used to prevent accidental delete by users and groups.
- Flag id value is:1

→ Create a directory under root (/) location:

```
#mkdir /imp-data  
#cd /imp-data  
#touch file1 file2 file3  
#ls
```

→ Setting full Permissions of a directory:

```
#chmod 777 /imp-data  
#ls -ld /imp-data
```

→ To enable sticky-bit for a directory:

```
#chmod +t /imp-data (or) #chmod 1777 /imp-data  
#ls -ld /imp-data
```

NOTE: The permission set is noted by the **lowercase t**, where the **x** would normally indicate the execute privilege.

→ Now login and test with any one normal user:

```
#su - raju  
$cd /imp-data  
$ls
```

→ Trying to remove any one file in the sticky-bit enabled directory:

```
$rm -i file1
```

NOTE: Getting an error **Operation not permitted.**

```
$exit
```

→ To disable sticky-bit for the enabled folder:

```
#chmod -t /imp-data (or) #chmod 0777 /imp-data  
#ls -ld /imp-data
```

Note that, now you are able to remove a file in the directory.