1. Execute
2. Write
3. Execute+write
4. Read

File Permissions describe the allowed operations by various users.

With respect to file permissions, all users are categorized into the following 4 types.

User Categories:

user/owner     Represented by 'u'

group   Represented by 'g'

 others     Represented by 'o'

all     Represented by 'a'

**Permission Types:**

**Number    Permission**

**0  No permission**

Execute

Write

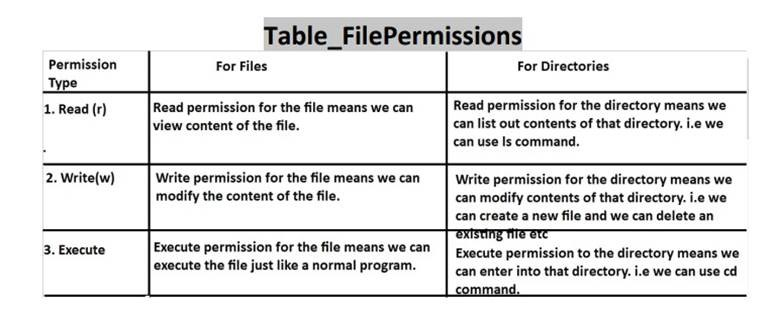
Execute and Write

Read

Read and Execute

Read and Write

Read, Write and Execute



Operations related to permissions:

We can perform the following 3 operations.

+ Add a particular permission to user|group|other|all

-    Remove a particular permission to user|group|other|all

= Assignment a particular permission to user|group|other|all

**. chmod Command:**

chmod means change mode.

We can use chmod command to change file or directory permissions.

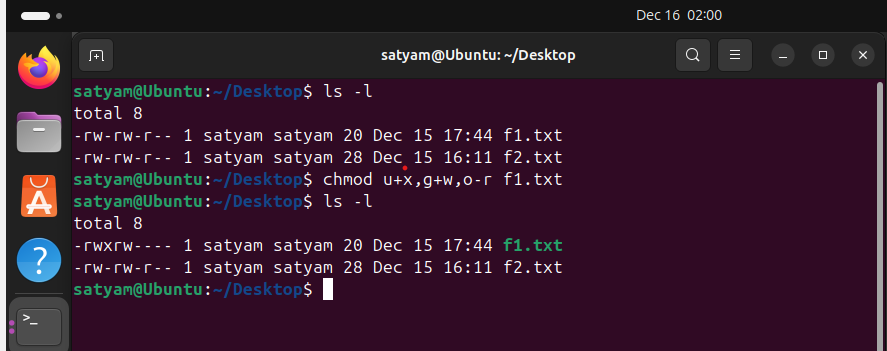
Syntax: $ chmod <user\_category><operation><permission> file\_name/directory\_name

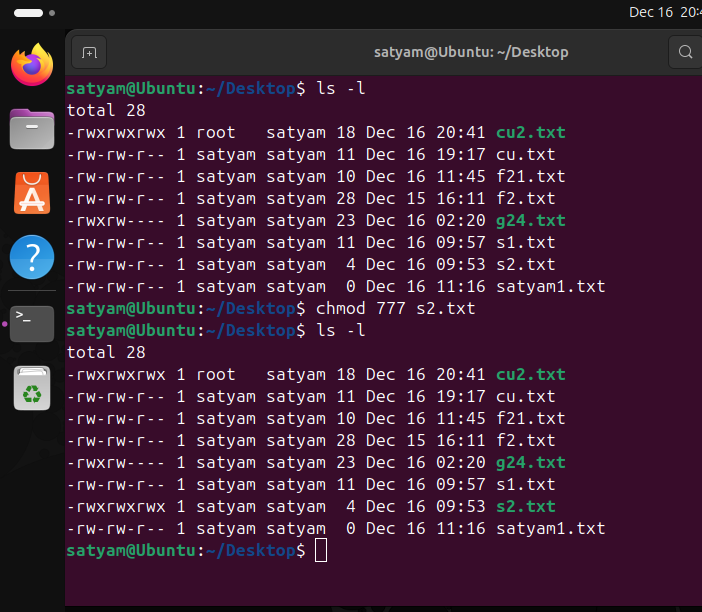
Eg: For user add execute permission,for group add write permission,for others remove read permission

$ chmod u+x,g+w,o-r demo.txt

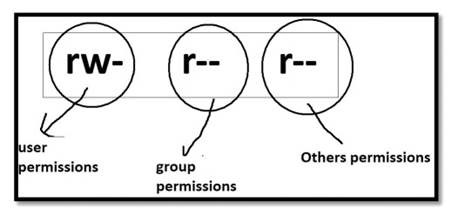
Note: Only owner and super user (root) can change file permissions.

How to check Permissions of existing File: By using ls -l command:





The file permissions are



Total 9 permissions. First 3 are user permissions, next 3 are group permissions and next 3 are others permissions.

user permissions: rwuser can perform both read and write operations but not execute operation

group permissions: r--

group members can perform only read operation and cannot perform write and

execute operations

others permissions: r-other members can perform only read operation and cannot perform write and execute operations.

User Permissions + Group Permissions + Others Permissions order is important

Eg 1: $ chmod u+x demo.txt adding execute permission to the user

Eg 2: $ chmod u+w,g+rw,o+r demo.txt adding write permission to the user adding read and write permissions to the group adding read permission to the others

Eg 3: $chmod u+x,g-w,o+w demo.txt adding execute permission to the user removing write permission from the group adding write permission to the others

Eg 4: $ chmod u=rw,g=rw,o=r demo.txt

Now user permissions: rwgroup permission: rwothers permission: r--

Eg 5: $ chmod a=- demo.txt

Now user permissions: --group permission: --others permission: ---

Eg 6: $ chmod a=rwx demo.txt

Now user permissions: rwx group permission: rwx others permission: rwx

**Read Permission to the File:-**

If the file not having read permission then we are not allowed to view content of the file. Hence cat, head, tail, more, less commands won't work.

**Write Permission to the File:-**

If the file not having write permission, then we cannot modify the content of the file.

**Execute Permission to the File:-**

If the user not has executed permission on any file, then he cannot execute that file as a program.

**Read Permission to the Directory:-**

If the user has read permission on any directory, then he can list out the contents of that directory. i.e he can use ls command.

**Write Permission on the Directory:-**

If the user has write permission on any directory, then he is allowed to modify the content of that directory. i.e he can add new files and remove existing files.

**Execute Permission to the Directory:-**

If the user not has executed permission on any directory, then he is not allowed to enter into that directory. i.e he cannot use cd command.

**Note:-**  If the user not having read permission on any file, then he cannot execute that file even though he has executed permission.

**Linux vs Security:**

The virus files usually created by others.

others are not having execute permission on our directories. Hence others are not allowed to add virus files to our directories.

Hackers are not having executed permission on our directories. Hence they cannot read our file data.

Because of this, Linux is considered as more secured operating system. Linux follows 2 levels of security.

1st level: login with credentials

2nd level: File and Directory permissions

**Note:**We are using permission types as r,w,x and these are considered as symbolic permissions. But we can also specify permissions by using octal number, such type of permissions are called numeric permissions.

**Numeric Permissions:-**

We can specify permissions by using octal number.

Octal means base-8 and allowed digits are 0 to 7.

          0  000     No Permission

         1    001     Execute Permission

          2    010     Write Permission

         3     011     Write and execute Permissions

         4     100     Read Permission

         5     101     Read and execute Permissions

         6     110     Read and write Permission

         7  111     Read, Write and execute Permissions

**Note:**

4 Read Permission

2 Write Permission

1 Execute Permission

It is more easy to remember

    5     4+1     r-x

    3     2+1     -wx

    6     4+2     rw

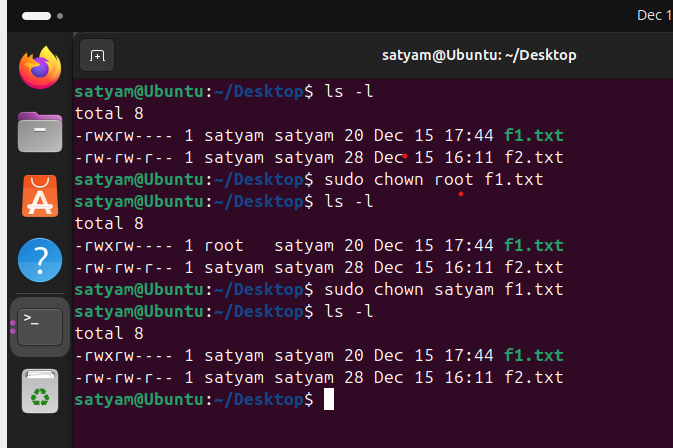
**chown :-**

chown means change owner.

Only root user can perform this activity.

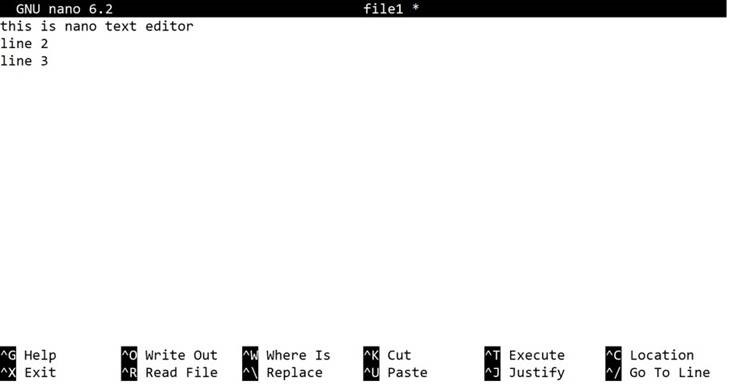
chown root demo.txt

Now the owner of demo.txt is root.



**. nano :-**

It is command line editor.It can be used to create new files and edit content of existing files



ctrl+g (F1) Display this help text

 ctrl+x (F2) Close the current file buffer / Exit from nano

 ctrl+o (F3) Write the current file to disk

 ctrl+r (F5) Insert another file into the current one

 ctrl+w (F6) Search forward for a string or a regular expression

ctrl+\ (M-R) Replace a string or a regular expression

ctrl+k (F9) Cut the current line and store it in the cutbuffer

 ctrl+u (F10) Uncut from the cutbuffer into the current line

But main important options:

ctrl+o To save content

 ctrl+x To quit from the editor