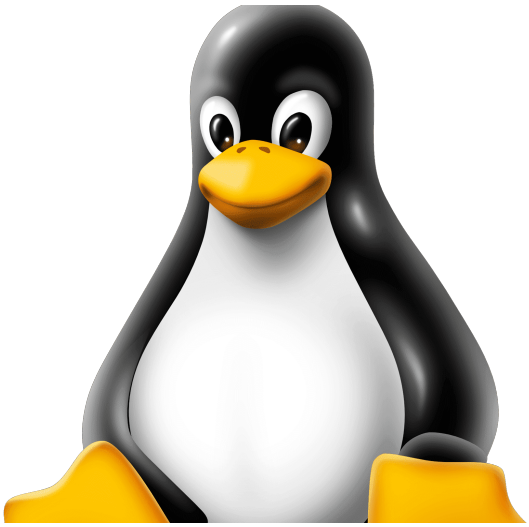


Linux

File System Permissions 🔐

Folder 📂

File 🧾

Read 📖 `R`

Read 📖 `R`

- List folder contents `ls folder`.

Write 📝 `W`

- Add files or folder inside folder `mv *.txt ~/files`.
- Delete, Move, Rename folder `rm | mv folder folder2 | folder ~/folders`.

Execute 🏃 `X`

- Go into the folder `cd folder/folderChid`.
- Read file contents `cat file.txt`.

Write 📝 `W`

- Edit, Delete, Move, Rename File `nano | rm | mv newFile.txt file.txt`.

Execute 🏃 `X`

- Run File as **script** like bash File `file.sh`.

❔ Why Permission types `R-W- X`

- To control of multi-user working on the same machine who have right to open that or modify it or deleted it or just read it.

Types of users 👥

1. `u` **Owner user**
    a. File creator

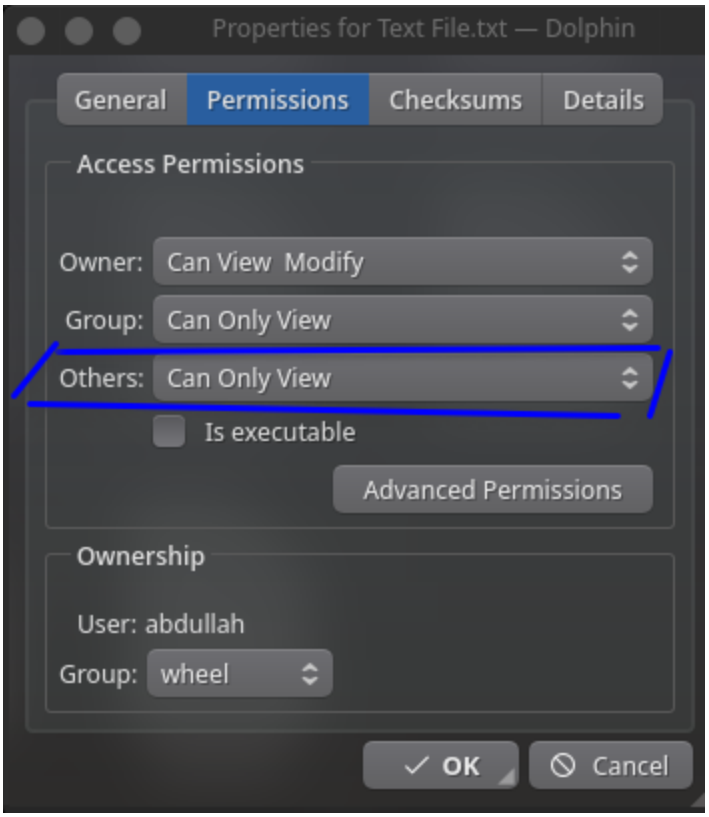
2. `g` **Group**

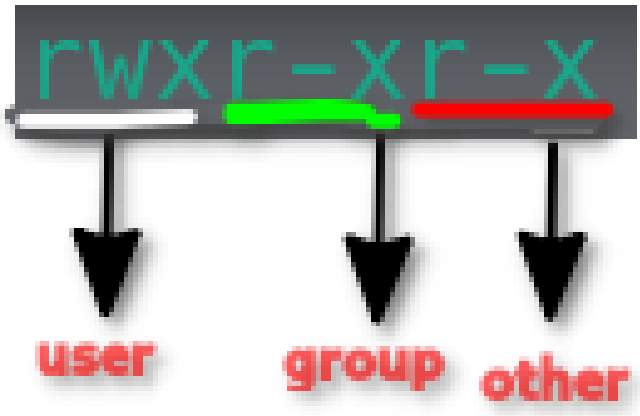
   a. Group of users have same permission.

   b. Instead of assign permission to each user individually.

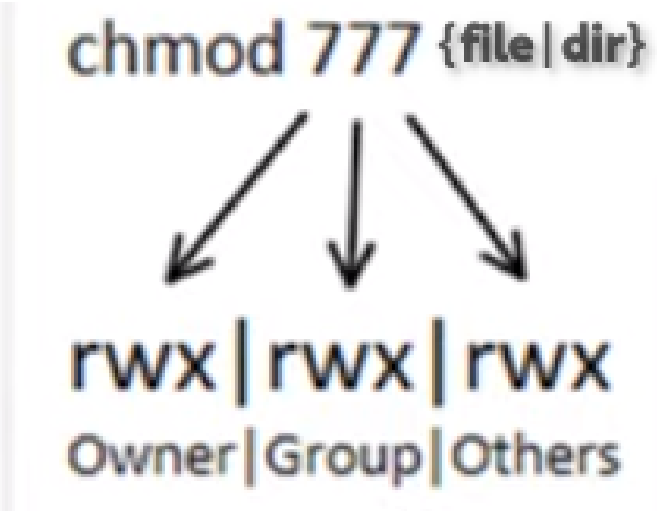
3. `o` **others** `all`

   a. Any user who have access to the file. `Not user nor group`





Change permission 🔓



There 2 ways

1. **Numerical or Absolute**

Numeric Value	Permission
0	---
1	--x
2	-w-
3	-wx
4	r--
5	r-x
6	rw-
7	rwx

```
$chmod 750 /projects/www/images #chmod {sum.user|sum.group|sum.other}
```

2. **Character or symbolic**


```
$chmod go+rwx file # chmod {ugoa|+-=|rwx} file
```

Wildcards

- It’s shell feature that made commands so powerful when using with wildcard.
- It’s Just some patterns to match something in your current directory or more.
    - Something list of files with specific pattern, etc.
- There are special characters.

Wildcard	Meaning

Wildcard	Meaning
*	Matches any characters
?	Matches any single characters
[characters]	Matches any character that is member of the set characters - Can be also expressed as POSIX character class such as one of the following: 1. `[:alnum:]` Alphanumeric characters 2. `[:alpha:]` Alphabetic characters 3. `[:digit:]` numerals 4. `[:upper:]` Uppercase alphabetic characters 5. `[:lower:]` Lowercase alphabetic characters
[!characters]	Matches any character that is not member of the set characters

Examples of wildcard matching

Pattern	Matches
`*`	All filenames
`g*`	All filenames that begin with the character "g"
`b*.txt`	All filenames that begin with the character "b" and end with the characters ".txt"
`Data???`	Any filename that begins with the characters "Data" followed by exactly 3 more characters
`[abc]*`	Any filename that begins with "a" or "b" or "c" followed by any other characters
`[[:upper:]]*`	Any filename that begins with an uppercase letter. This is an example of a character class.
`BACKUP.[[:digit:]][[:digit:]]`	Another example of character classes. This pattern matches any filename that begins with the characters "BACKUP." followed by exactly two numerals.
`*[![:lower:]]`	Any filename that does not end with a lowercase letter.

Examples of wildcard matching

cp

- It’s program copies files and directories.
- In its simplest form, it copies single file.

```
[me@linuxbox me]$ cp file1 file2
```

It can also be used to copy multiple files (and/or directories) to a different directory:

```
[me@linuxbox me]$ cp file... directory
```

A note on notation: ... signifies that an item can be repeated one or more times.

Other useful examples of `cp` and its options include:

Command	Results
`cp` *file1 file2*	Copies the contents of *file1* into *file2*. If *file2* does not exist, it is created; **otherwise, *file2* is silently overwritten with the contents of *file1*.**
`cp -i` *file1 file2*	Like above however, since the "-i" (interactive) option is specified, if *file2* exists, the user is prompted before it is overwritten with the contents of *file1*.
`cp` *file1 dir1*	Copy the contents of *file1* (into a file named *file1*) inside of directory *dir1*.
`cp -R` *dir1 dir2*	Copy the contents of the directory *dir1*. If directory *dir2* does not exist, it is created. Otherwise, it creates a directory named *dir1* within directory *dir2*.

Examples of the cp command