COMP311
Linux OS Laboratory
Lab4:File Systems (2)
(File Metadata)

By

Alaa' Omar

Objectives

1

Understand and manipulate permissions (mode) on different Linux files.

2

Set the default permissions for files and directories.

3

Identify and handle file properties such as ownership, groups,

4

size, and timestamps.

Permissions in Linux

Types of users?

1. **user (u)** = user (owner) permissions on the file (file creator).
2. **group (g)** = permissions of members of the group name stamped on the file (except owner), group of users have the same permissions, instead of assign permission to each group member individually, we can add them to a group.
3. **other (o)** = all system users other than the group and owner.

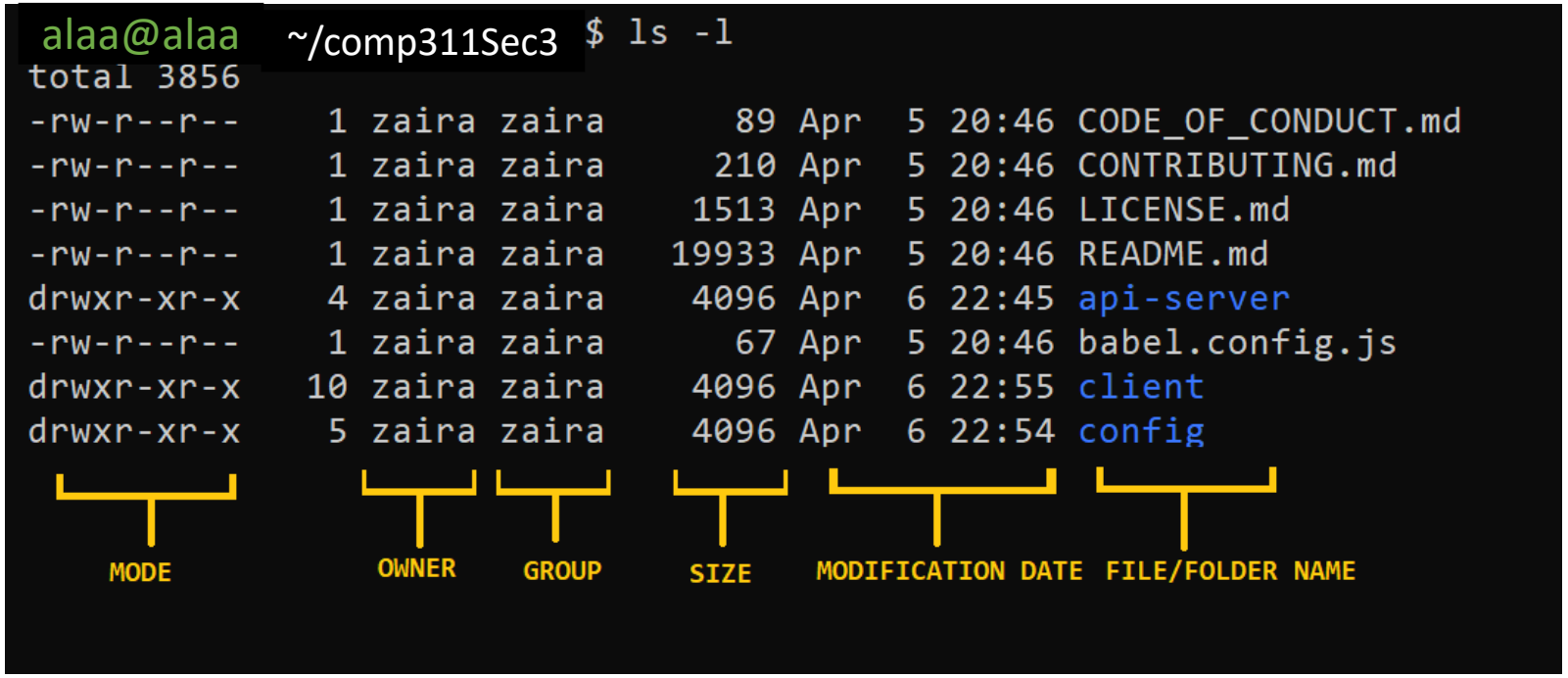
Each file has nine characters that represent the permissions on that file. Those are divided into three equal parts:

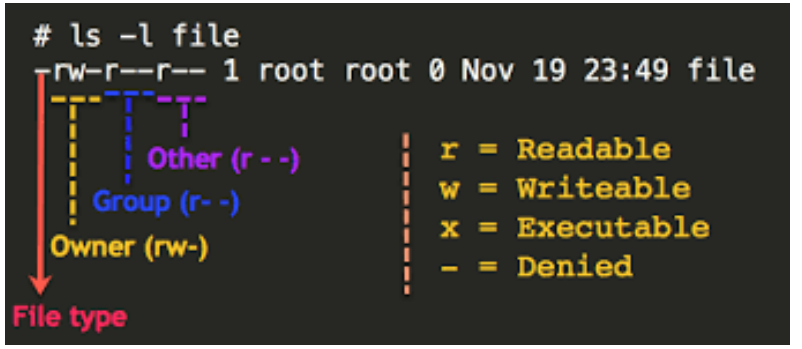
Ordinary File	
Read (r)	Read from the file using vi, more, cat, …)
Write (w)	Write to the file (modify the content)
Execute (x)	Execute the file, run the file (scripts or binaries)
Directory	
Read (r)	Read the directory (using ls)
Write (w)	Create/remove subdirectories and files inside it.
Execute (x)	The user can access the directory (use cd).

Permission Linux Commands

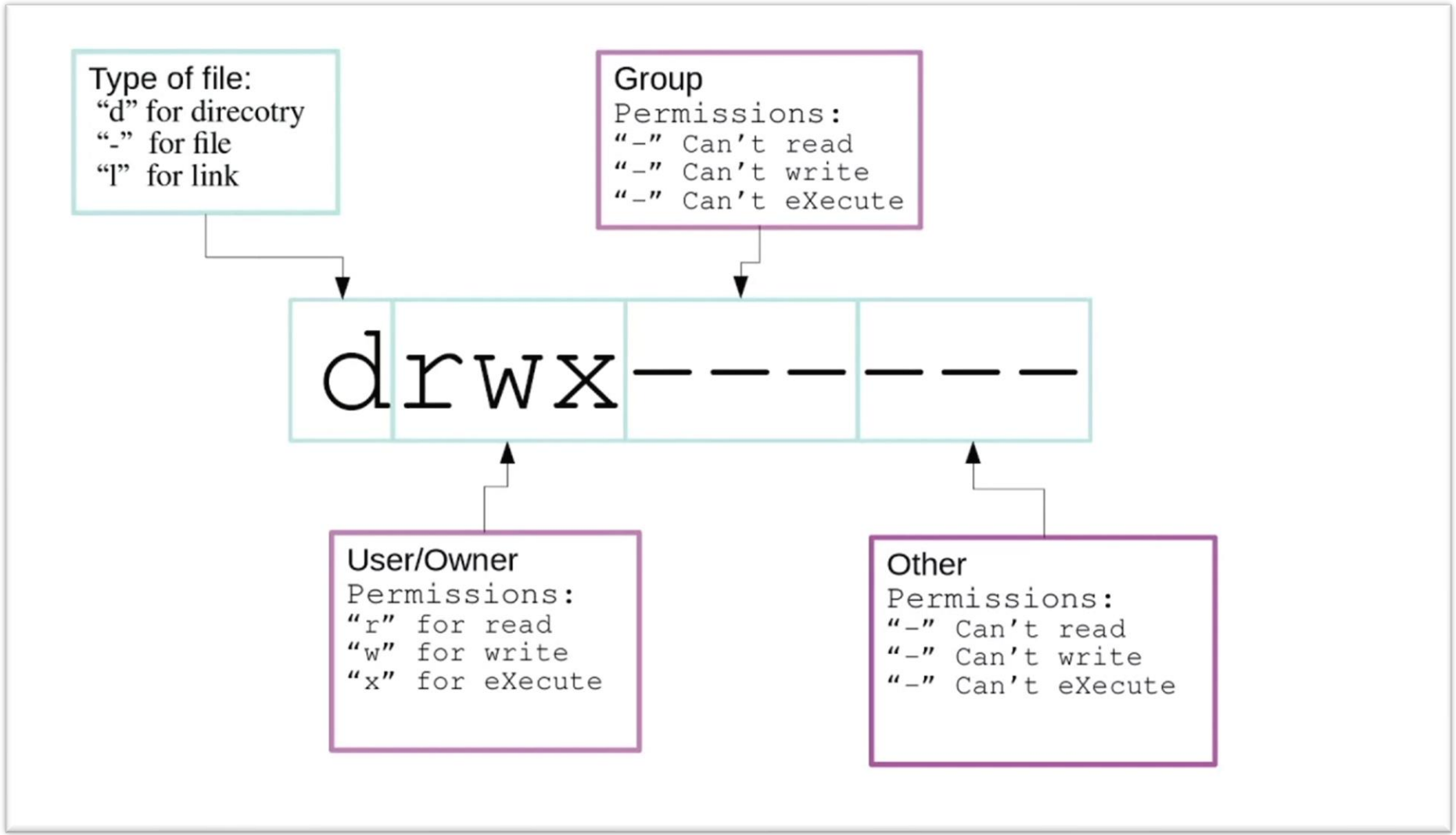
To view the permissions of a file or a directory we use the ls -l command

$ls -l file Where -l stands for long format





Permission Anatomy in Linux



Permission Linux Commands

To change the mode, a user may use the chmod (change mode) command. This command can specify the new permissions using a relative or absolute method.

In your terminal

Type $man chmod

$chmod mode file

Where :

- **mode -> the new file mode**
- **file -> the name of the file**

alaa@DESKTOP-CEO72GJ: ~

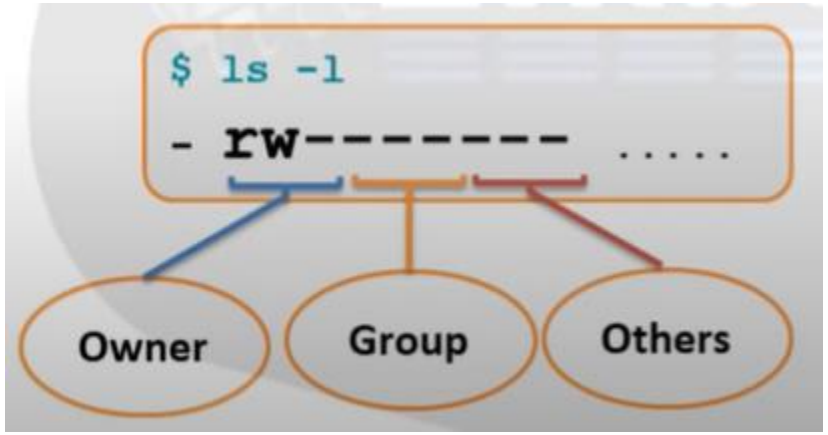
```
CHMOD(1)                                  User Commands                                  CHMOD(1)

NAME
       chmod - change file mode bits

SYNOPSIS
       chmod [OPTION]... MODE[,MODE]... FILE...
       chmod [OPTION]... OCTAL-MODE FILE...
       chmod [OPTION]... --reference=RFILE FILE...

DESCRIPTION
       This  manual page documents the GNU version of chmod.  chmod changes the file mode bits of each given file ac-
       cording to mode, which can be either a symbolic representation of changes to make, or an octal  number  repre-
       senting the bit pattern for the new mode bits.

       The  format  of  a  symbolic mode is [ugoa...][[-+=][perms...]...], where perms is either zero or more letters
       from the set rwxXst, or a single letter from the set ugo.  Multiple symbolic modes can be given, separated  by
       commas.

       A  combination of the letters ugoa controls which users' access to the file will be changed: the user who owns
       it (u), other users in the file's group (g), other users not in the file's group (o), or all  users  (a).   If
       none  of  these  are given, the effect is as if (a) were given, but bits that are set in the umask are not af-
       fected.

       The operator + causes the selected file mode bits to be added to the existing file mode bits of each  file;  -
       causes them to be removed; and = causes them to be added and causes unmentioned bits to be removed except that
       a directory's unmentioned set user and group ID bits are not affected.

       The letters rwxXst select file mode bits for the affected users: read (r), write (w), execute (or  search  for
 Manual page chmod(1) line 1 (press h for help or q to quit)
```

Chmod using relative method

Symbol	Description
u	User
g	Group
o	Other
a	All

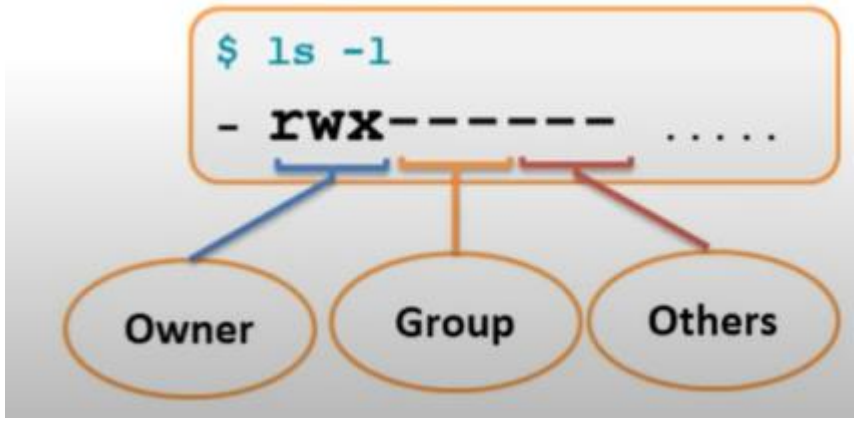
Symbol	Description
r	read
w	write
x	execute

Symbol	Description
+	Means add
-	Means subtract
=	Means set/overwrite



chmod u+x myfile

Add execute permission to the user (owner) of the file

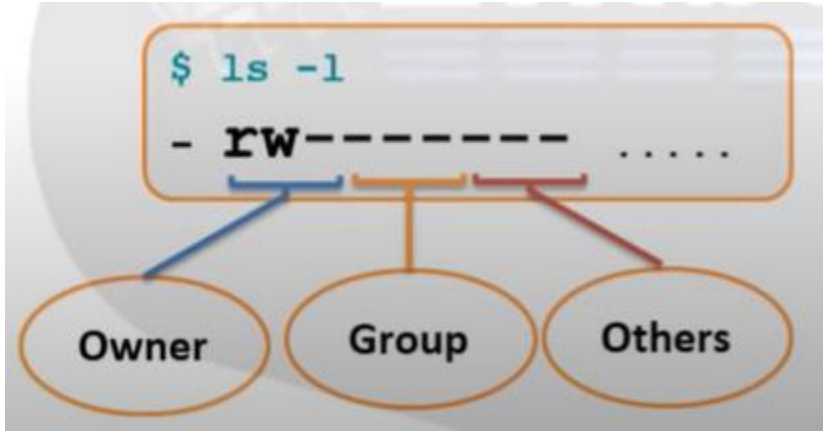


Chmod using relative method

Symbol	Description
u	User
g	Group
o	Other
a	All

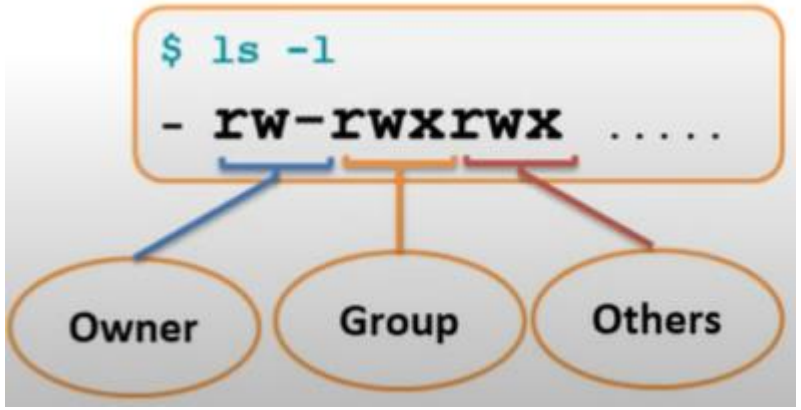
Symbol	Description
r	read
w	write
x	execute

Symbol	Description
+	Means add
-	Means subtract
=	Means set/overwrite



chmod go+rwx myfile

Add read, write, execute permissions to the group and others

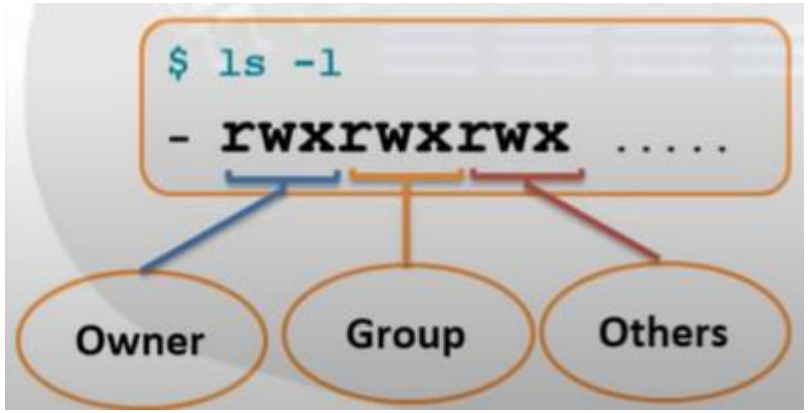


Chmod using relative method

Symbol	Description
u	User
g	Group
o	Other
a	All

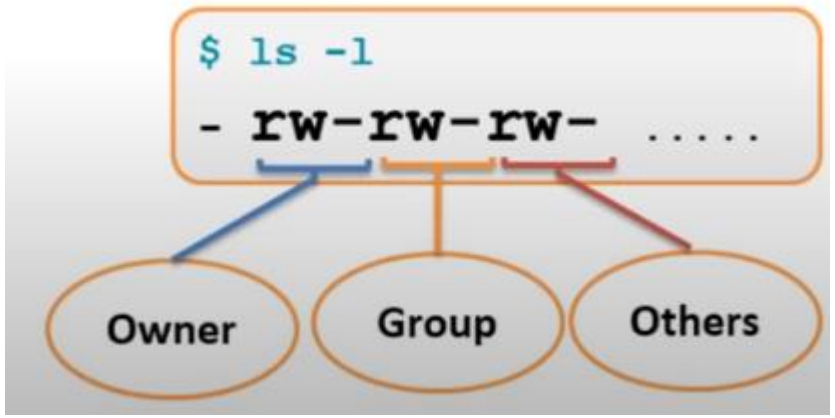
Symbol	Description
r	read
w	write
x	execute

Symbol	Description
+	Means add
-	Means subtract
=	Means set/overwrite



chmod a-x myfile

remove execute permission on the user, group and others

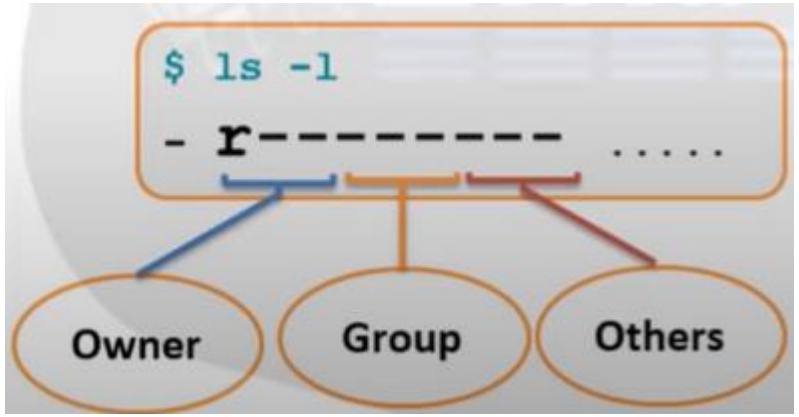


Chmod using relative method

Symbol	Description
u	User
g	Group
o	Other
a	All

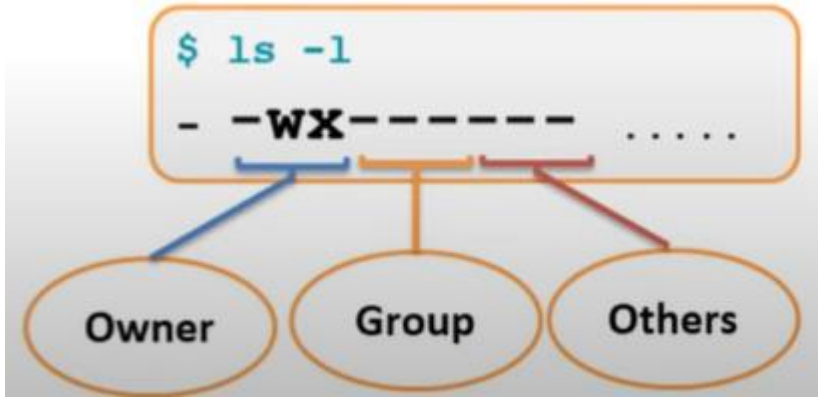
Symbol	Description
r	read
w	write
x	execute

Symbol	Description
+	Means add
-	Means subtract
=	Means set/overwrite



chmod u=wx myfile

Overwrite owner permission to write and excute



Chmod using relative method (summary)

- **Using this method, the user can modify the permissions on a file (or directory) relative to the already existing permission as follows:**
- **Add permissions**
  - **chmod u+rx,g+rw,o+r myfile**
- **Remove permissions**
  - **chmod u-wx, g-w,o-w myfile**
- **Add/remove permissions**
  - **chmod u-x+w,g-x+r,o-rw+x myfile**
- **Set (overwrite) permissions:**
  - **chmod u=rw, g=wx,o=r**

Chmod using relative method

- Using this method, the user can modify the permissions on a file (or directory) relative to the already existing permission as follows: Assume that we start with the following permissions on a file called

myfile ***r-xrw-r--***

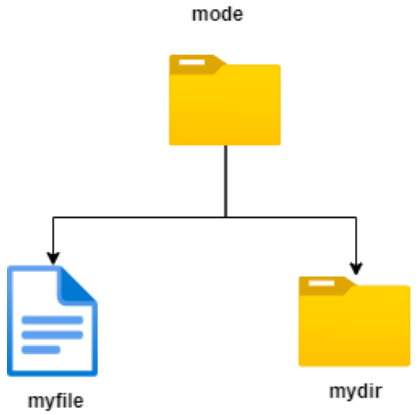
The command: **chmod u+w,g-rw,o+x myfile**

Will change the permissions on myfile to **rwx---r-x**

If we continue with the command: **chmod u=rw,g+w myfile**

The permissions will now become **rw--w-r-x**

Chmod using relative method (practice)



- Create a directory called **mode** and move inside it
- Create a file called **myfile** and a directory called **mydir** inside directory **mode**.
- Using the **chmod** command with relative mode, change the permissions on
- both **myfile** and **mydir** as follows:

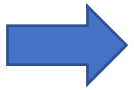
First run the command ls –l mode

- **rwxr-xrw- commands**
- **r--rw---x** commands=
- **---rwx-wx** commands=

Chmod using absolute method

- The second method is the absolute method which **does not depend on the permissions that already exist on the file**.
- This method uses a **binary 1** where you want a permission to be set and a **binary 0** where you want it unset

u (user)			g (group)			o (other)		
r	w	x	r	w	x	r	w	x
1	0	0	0	1	0	0	0	1
4			2			1		

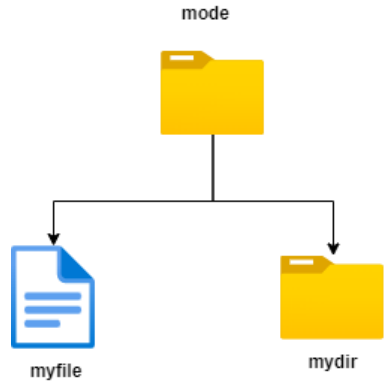


Permission	Binary	Octal
r	100	4
w	010	2
x	001	1

Permission	Binary	Octal	
x	001	1	1
w	010	2	2
r	100	4	4
wx	011	2+1	3
rx	101	4+1	5
rw	110	4+2	6
rwx	111	4+2+1	7

Chmod using absolute method (practice)

- Using the **chmod** command with absolute mode, change the permissions on both **myfile** and **mydir** as follows:
- **rwxr-xrw- commands= _____;_____.**
- **r—rw---x commands= _____;_____.**
- **---rwx-wx commands=_____;_____.**



Hint:
To avoid problems, don not give execute permission to a file that is not executable.

u (user)			g (group)			o (other)		
r	w	x	r	w	x	r	w	x
1	0	0	0	1	0	0	0	1
4			2			1		

Permission	Binary	Octal
r	001	4
w	010	2
x	100	1

Permission	Binary	Octal	
x	001	1	1
w	010	2	2
r	100	4	4
wx	011	2+1	3
rx	101	4+1	5
rw	110	4+2	6
rwx	111	4+2+1	7

How UNIX assign permissions to a new file: umask

umask → user mask

When linux creates a new file, it starts with a file mode of:

777: for excitable ordinary files

777: for directories

From this initial mode, Unix subtracts the value of the USER MASK. The user mask is a mode, set by you, showing which permissions you want to restrict.

Default Mode

- The default permissions that are set on newly created files and directories are set using the **umask** command.
- Run the command:
  - **$umask**
- What number did you get → 0022 .
- Type **$man 2 umask**
- Expected permissions on a new file=_____.
- Expected permissions on a new directory= _____.

Umask – How to get permission

- umask 022

(subtract)

initial mode	7	7	7
umask mode	0	2	2
permission	7	5	5
	4+2+1	4+1	4+1
	rwx	r-x	r-x

This user mask shares your files without letting Anyone change them.

File permission: rw- r– r--
Directory permission: rwx r-x r-x

- umask 077

(subtract)

initial mode	7	7	7
umask mode	0	7	7
permission	7	0	0
	4+2+1	0	0
	rwx	---	---

This user mask withhold all permissions (read, write and execute) from your group and from anyone else.

File permission: rw- --- ---
Directory permission: rwx --- ---

Question: If you want to change the file permission to r--rw- -w-. List at least 3 masks that should be used to get permission for this file:

- Step (1): Change permission to numeric method, in our case it will be 462
- Step(2): calculate the mask by subtracting the permission from 777, in this case 777 -462 = 315. This is the first umask.

- Based on the fact that the file execution permission is not set by default, regardless giving it using umask, you can paly with the execution bit in the three part and it will give you different umasks but the same permission.
- Step(3), paly with the permission by adding or removing x permission. For example, r-xrw—w- → 562 →777-562=215. This is the second umask.
- r—rwx-w- → 472→ 777-472=305.

Can you find another umasks?!!!!

umask (practice)

- What permissions would you expect after the command: ***umask 625*** is executed. Try it to see the results. Did it work?_____.
- Now let us try and do the reverse:
- ***If you want a newly created directory to have the permissions rwxr---wx what umask command would you run: _____.***
- ***Try it. Did it work? _____.***
- ***To have the following permissions on a newly created file: r---rw---w- what umask command would you run:_____.***
- ***Try it. Did it work?_____.***
- ***What about if you wanted a newly created file to have permissions: rwxr---wx. What umask command would you run: _____.***
- ***Try it. Did it work? _____. Why? _____.***

Changing Link Properties

- Since we can modify the mode property of a file, we can do more testing to see how links work. Go back and create two files called **file1** and **file2** and then create a hard link called **hlink** to **file1** and a symbolic link called slink to file2. List the commands you used:

- **$touch file1 file2**
- **$ln file1 hlink**
- **$ln –s file2 slink**

Change Link properties (Practice)

- ***Now try changing the permissions on file1 to* rwxrwxr--*. Command:***
- ***What happened to the permissions on hlink? Why?***
- ***Now change the permissions on hlink to* rwx------x.**

Change Link properties (Practice)

- ***Command:_____.***
- ***What happened to the permissions on file1? Why?***
- ***_____.***
- ***Now try changing the permissions on file2 to rw_r_xr__.***
- ***Command:_____***
- ***What happened to the permissions on slink? Why?***
- ***_____.***
- ***Now change the permissions on slink to r__rwxr_x.***
- ***Command:_____.***
- ***What happened to the permissions on file2? Why?***
- ***_____.***
- ***What happened to the permissions on slink?_____.***

Ownership and Groups

- The next file property is the name of the owner of the file. The owner is the only user (other than root) that can modify the properties of a file. The root is the only one that can change a file ownership using the command ***chown*** as follows:
- **chown newuser filename**
- ***Try changing the ownership of any of your files. Did it work?_____.***
- The following file property is the group name on the file. This group name may be modified by the owner if he/she is a member of the new group he/she wants to put on the file. To change the group, a user uses the command ***chgrp*** as follows:
- **chgrp newgroup file**
- ***Try to change a group on any of your files. What happened?_____.***

Size

- The next property shows the size (in bytes) of a file. Try creating a file and putting the phrase “how are you” inside then save and quit. **What is the size of the file?_____.**
- **Why?_____.**
- **Change to directory /dev. Command:_____.**
- **Check out the size property on device files. What did you find?**
- **_____.**

Size

- ***What are the two numbers that exist instead of the size?***
- ***_____***
- ***_____.***
- ***Go back to your home directory. Command:_____.***
- ***Go back and display the size of the symbolic link (slink) you created earlier. Can you figure out how that size was calculated?_____.***
- ***Try creating a new symbolic link and see if you are able to figure out how the size on a symbolic link is set. What did you find?***
- ***_____.***

Size /dev files

- If you issue the *ls -l* command, you'll see two numbers (separated by a comma) on the device file entries before the date of last modification, where the file length normally appears. These numbers are the **"major"** and **"minor"** numbers for the device. The following listing shows a few devices as they appear on a system. Their major numbers are 10, 1, and 4, while the minors are 0, 3, 5, 64-65, and 128-129.

```
crw-rw-rw-   1 root     root      10,   3 Nov 30  1993 bmouseatixl
crw-rw-rw-   1 root     sys        1,   3 Nov 30  1993 null
crw-rw-rw-   1 root     root       4, 128 Apr 30 13:02 ptyp0
crw-rw-rw-   1 root     root       4, 129 Apr 30 13:02 ptyp1
crw-rw-rw-   1 rubini   staff      4,   0 Jan 30  1995 tty0
crw-rw-rw-   1 root     tty        4,  64 Jan 25  1995 ttyS0
crw-rw-rw-   1 root     root       4,  65 May  1 00:04 ttyS1
crw-rw-rw-   1 root     sys        1,   5 Nov 30  1993 zero
```

Time Stamps

- A file has several time stamps. The main two are:

1) **Last modification time**: which is the time the file was last modified and saved. This is the default time displayed by **ls –al** command.
2) **Last access time:** which is the time the file was last accessed or viewed. What ls option is used to display that time. _____ (Check the man pages).

In Linux, it is easy to look inside the inode for a particular file by using the stat command, just type stat followed by The file name **$ stat filename**

Time Stamps

- Check the times on file ***myfile*** and record them.
- Now view the file using the ***more*** command***. What happened to the times?***
- _____***.***
- Now open the file ***myfile***, modify it and then save and quit.
- ***What happened to the times now?***
- _____***.***
- Another way to display file properties in detail is to use the ***stat*** command. Run the ***stat*** command on file ***myfile*** as follows:
- ***stat myfile***
- ***What information can you see:***

_____

_____

- *For more information on the output, you can read the man pages on the* ***stat****.*

File name

- A Linux file name can be up to 255 characters long and is made of any characters. A dot has no special meaning in a file name except if it is the first character then the file is a hidden file. ***Create a hidden file called .hidden. Command:***
  - ***Try to list your files using the command ls. Can you see .hidden?***
- ***Now try to list the files using the command ls with the –a (all) option? Can you see it now?***

The End