Lab 4. File Systems (II) (File Metadata)

Comp311- Lab Linux

Instructor :Murad Njoum

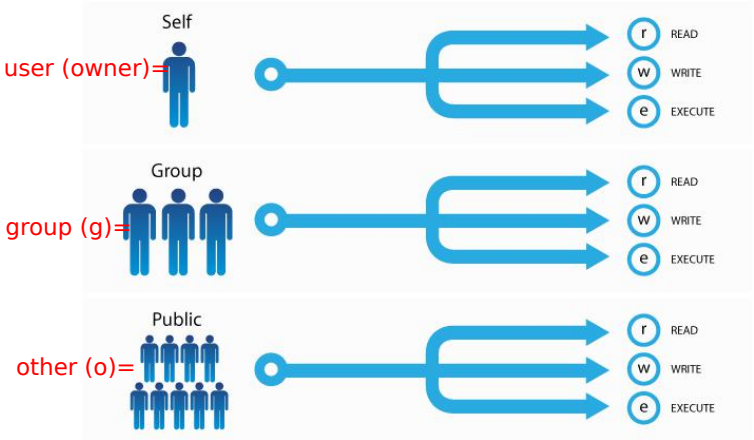
Objectives

After completing this lab, the student should be able to:
- Understand and manipulate permissions (mode) on different Linux files
- Set the default permissions for files and directories
- Identify and handle file properties such as ownership, groups, size, and timestamps.

Permissions (Mode)

Each file has nine characters that represent the permissions on that file. Those are divided into three equal parts:
user (u)= user (owner) permissions on the file.
group (g) = permissions of members of the group name stamped on the file (except owner).
other (o)= all system users other than the group and owner.
The main three permissions that may exist on the file are ***read (r), write (w),*** and ***execute(x)***.

Linux File Systems



These mean different things for files than they do for directories as follows:

Read: for **files** it means the user can **view content of file** (using vi, more, cat, ...) while for **directories** it means the user can view content of directory (**using ls**).

Write: for **files** it means the user can **modify the content of the file**, but for directories it means the user can modify the content of the **directory** (i.e. can create or remove files and subdirectories).

Execute: for **files** it means the user can run the file (**scripts or binaries)** while for **directories** it means the user can access the directory **(use cd)**

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.

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**

Check the man pages on command ***chmod*** for more examples and then do the following

Absolute and Relative Paths

- Create a directory called ***mode*** and move inside it (***mkdir mode; cd mode***).
- 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:

Example :rwxr-xrw- commands=

chmod _____**;** **chmod** _____**.**

On/Off

Other Example

chmod _____**.** **chmod** _____

Note :Capital letter not allowed

Try chmod gou=rwx file1
Try chmod ugo=- file1
Note : be careful when use – sign, try chmod u=r-xw , make sure rwx is on

Practice:

First of all run command chmod u=rwx,g=rwx,o=rwx file1 dir1 , then try to answer the exercise? For both file and directory

A. r-- rw- --x commands=

_____**;**_____**.**

_____**;**_____**.**

B. rwx-wx-- commands=

_____**;**_____**.**

_____**;**_____**.**

Deep Thinking: *What does the following records mean ? Your Notice?*

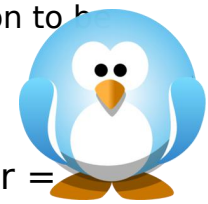
Remove directory mode, then recreate it and inside it create two directories and Two files also. You should have something like this? Now answer the above question?

```
drwxrwxr-x  4 mnjoum mnjoum  4096 Oct  7 21:54 .
drwx------ 72 mnjoum mnjoum 32768 Oct  7 21:39 ..
-rw-rw-r--  1 mnjoum mnjoum     0 Oct  7 21:39 file1
-rw-rw-r--  1 mnjoum mnjoum     0 Oct  7 21:39 file2
drwxrwxr-x  2 mnjoum mnjoum  4096 Oct  7 21:54 mydir1
drwxrwxr-x  2 mnjoum mnjoum  4096 Oct  7 21:54 mydir2
```

Line 1: permissions for current directory
Line2: Permissions for parent directory
Line3 and 4: Permissions for file 1 and file 2
Line 5 & 6: Permissions for mydir1 & mydir2
Note :permissions for both file1 and file2 are same , mydir1 & mydir2 also are same
(default permissions for file, later we will speak in more Details)

Absolute Method:

does not depend on the permissions that already exist on the file.
This method uses a binary **1** where you want a permission to b set and a binary **0** where you want it unset as follows:



The command: **chmod 734 file**
will set the permissions on **file** to 111 (7) for user = rwx and 011(3) for group = -wx and 100(4) for other = r-- so the permissions on the file will be= **rwx- wxr--**

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=

_____;_____.

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:_____.

This number decides the permissions **set on newly created files or directories.**

Read the manual page for umask (***man 2 umask***) and try to figure out what permissions would you get on files and directories after you run the command: **umask 123**

Expected permissions on a new file=

_____.

Expected permissions on a new directory=

_____

Important:

Full Permission for directories is 777, But when we create directory (mkdir), the permission is 755

WHY

755:since, **group and others** haven't to write on user directory
Cann't create or remove files and subdirectories

Full Permission for files is 666, But when we create directory (vi, touch), the permission is 644

WHY

666:Have no executions (run scripts) for files

Important:

When you are dealing with UMASK *you should keep two things in mind , the full permission of an object is 777 and we cannot set execution permission for a file using* **UMASK**.

How UMASK works.?

*Subtract the umask value from 777, result will be the permission of the directory and if the result contains any execution permission, **exclude the execution** bit and it will be the permission of the file . Note, however, that files are not usually created with the execute permission by default, so the final permissions for files will omit the "x" permission.*

How the calculation should be done

For example your UMASK is 222

```
777 -  Full permission
222    UMASK value
-------------------------------
555    =>  This will be the permission of the Directory.
555    =>   r-x,r-x,r-x
```

Important:

The result contains execution permission. So exclude x bit from the result, it will be the permission of the file. r--,r--,r-- => 444
Final result -: If UMASK is 222 then **directory's permission** will be 555 and **file's permission** will be 444.

For example your UMASK is 333

```
777  - Full Permission
333   UMASK value
--------------------------------
444   =>  This will be the permission of the Directory.
444   =>  r--,r--,r--
```

The result contains no execution permission. So no need to exclude x Bit.
Final result -: If UMASK is 333 then directory's permission will be 444 and file's permission will be 444.

Important:

For example your UMASK is 666

```
777  - Full Permission
666   UMASK value
--------------------------------
111   =>  This will be the permission of the Directory.
111   =>  --x,--x,--x
```

The result contains only execution permission. So if we exclude them there will be nothing left for the file permission and it will be blank.
Final result -: If UMASK is 666 then directory's permission will be 111 and file's permission will be 000.

Umask:

Check to see if you understood how **umask** works by creating a new file and a new
directory and checking the set permissions. Did you get it right? _____.
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-- --x. 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:

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.
_____

Changing Link Properties:

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?_____.

Note: automatically permissions changed on hfile1 and vis versa is true File2 not changed (vis versa) directory & file system on other devices have not be accessible.

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/s
wants to put on the
file. To change the group, a user uses the command **chgrp** as follo

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?
_____

/dev$ ls -l sda*

```
mnjoum@ubuntu:/dev$ ls -l sda*
brw-rw---- 1 root disk 8, 0 Oct  8 11:08 sda
brw-rw---- 1 root disk 8, 1 Oct  8 11:08 sda1
brw-rw---- 1 root disk 8, 2 Oct  8 11:08 sda2
brw-rw---- 1 root disk 8, 3 Oct  8 11:08 sda3
brw-rw---- 1 root disk 8, 4 Oct  8 11:08 sda4
brw-rw---- 1 root disk 8, 5 Oct  8 11:08 sda5
brw-rw---- 1 root disk 8, 6 Oct  8 11:08 sda6
brw-rw---- 1 root disk 8, 7 Oct  8 11:08 sda7
```

What are the two numbers that exist instead of the size?
_____
_____.

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?

_____.

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**).
Check the times on file **myfile** and record them.

Now view the file using the **more** command***. What happened to the times?***

Time Stamps :

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**.

Access: Last time file was accessed (read)
Modify: Last time file modified
Change: last time file metadata changed (specially for permissions)

$ stat myfile
Access: 2018-10-08 12:10:26.613659929 +0300
Modify: 2018-10-08 12:10:26.613659929 +0300
Change: 2018-10-08 12:10:26.649660413 +0300

$ more myfile

$ stat myfile
Access: 2018-10-08 **12:14:31**.396921117 +0300
Modify: 2018-10-08 12:10:26.613659929 +0300
Change: 2018-10-08 12:10:26.649660413 +0300

chmod +x myfile

$ stat myfile
Access: 2018-10-08 12:14:31.396921117 +0300
Modify: 2018-10-08 12:10:26.613659929 +0300
Change: 2018-10-08 **12:16:14**.966281526 +0300

Try to modify the file content, what happened to Access, Modify, Change?

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?_____

Thank You for attention !

Published By: Murad Njoum