Welcome
Linux 2022

www.adafrolabs.com

One of the jobs of the operating system is to maintain a set of access permissions for each file present on the file system. Linux and other Unix-like operating systems separate permissions of 3

different entities:

- ☐ File user (synonymous with owner)
- ☐ Group
- ☐ Others

If you look at the long output of the Is command, current permissions are shown as a series of symbols (flags), most often being

r	= read, allowing you to view and copy a file
w	= write, allowing you to change and potentially delete a file
x	= execute, allowing you to run a program such as a shell script

The permissions shown by the is command are positional, as letters such as r, w or x are present when a given permission is set, or a hyphen/dash is shown when permission is not set.

Permissions are set by the file owner using the **chmod** command. The chmod command allows you to specify the desired permissions in two ways:

Method #1: Numerical method, using octal values

r has the value of 4

w has the value of 2

x has the value of 1

Permissions are set to all the entities (user/group/others) at once, with permissions for a given entity specified as a numerical sum of the desired letters for each entity (user, group and others),

for example:
permissions "rw- r-x r--" would be set using the command: **chmod 654** filename while chmod 123 filename would result in permissions --x-w--wx w

While the numerical method has its use, it has a major disadvantage that every time used, all 3 entities are re-set which forces the user to make constant calculations (albeit simple) if they wish to set some permissions while preserving others.

Keep in mind that the numbers are specified in the octal numbering systems and should be pronounced one digit at a time - for example 400 is read "four-zero-zero", not "four hundred".

- Create a new file called **example** in your home directory (using command touch example).
- Use the **ls -1 example** command to examine the default permissions. Now, using the **chmod 700 example** command, change the permissions for that file. Again, use the **ls -1 example** command to verify the results.
- Try setting other permissions and see how the ls -l output changes. Try deleting the file with the write flag removed.
- Spend some time practicing octal number addition, so you are comfortable with converting permission symbols to and from their numerical equivalent.

Method #2: Symbolic method, using letters and operator symbols

Compared to the numerical approach, this method offers more flexibility by allowing the user to change specific permissions (only read for example) for one OR more entities (user/group/others).

Symbolic method uses letters to specify entities:

u = user
g = group
o = others
a = all, refers to the three entities above.

Letters u, g and o can be used individually or combined, for example **go** allows you to set the same permissions for **group** and **others** at the same time.

Permissions r, w and/or x are set using +, - and/or = operators, for example:

chmod **u=rw,g=rx,o= somefile** is the same as **chmod 650 somefile**
while **chmod u+x somefile** will simply add the execute flag for user regardless of other permissions

Keep notes for this part in the space provided. Use only symbolic method to set the permissions in this section. Use the **ls -1** command to verify the outcome of each step.

- Create a new file in your home directory and note the default permissions.
- Remove all permissions to the above file.

Command #1: ______________________________

- Give yourself read and write access and read access to group and others.

Command #2: ______________________________

- Remove the read access for others and add execute permissions for yourself and the group.

Command #3: ______________________________

- Prevent yourself from easily deleting the file:

Command #4: ______________________________

Compare and discuss your notes with a classmate before moving on

Keep in mind that the purpose of the permissions is file security. You should use them to protect files you own and give limited access to these files to others as needed. Typically, write permissions for group and others are not desired as they may allow them to change or even delete files.
Avoid using a "7" permission for files, unless you are in fact referring to a program. Permission of "777" is used only for testing - it should not be used as anyone has full access this way.

Directory Permissions

Directories are special files and although the same letters r, w and x and the same command syntax are used to describe and set permissions as for files, they have a different meaning here:

r = read, allows you to view directories using commands such as ls and tree
w = write, allows you to change directory contents - create, rename and delete files inside and protect the directory itself from deleting it
x = pass-through, allows you to access the directory inside or use it as a part of a path

The pass-through permission controls access and essentially allows you to protect contents without the need to change permissions of every single file inside.

- Create a new directory in your home. Use the **ls -ld** command to view the default permissions and note how they differ from regular files. Do you understand why the "x" is needed for new directories?
- Without changing your PWD, create a new file in the new directory (you can use the **touch** command for that). Now, remove the write permission for the directory and try deleting (**rm** command) and re-naming (my command) the newly-created file. You should not be able to do that as the directory write permission controls that. Use the cd command to descend into the practice directory and list it. You can also view and change the content of files inside, as that is controlled by the individual file permissions.
- Now using the **cd ..** command leave the practice directory and change the permissions to **100** for it. Use the **ls -ld** command to verify that you did it correctly. Try listing the directory contents and notice the unusual output. The pass-through permission is sufficient to access the directory contents, even if it cannot be listed. You can access files inside as long as you know/remember their names.
  Try using an editor such as nano (nano your-dir/your-file) to make changes to the file content.

Always remember how directory permissions interact with permissions of files inside. Ultimately what can be done with a file is a combination of permissions of the file itself and the container directory.