Howkmii Academy

Version Control Class:

1) Warm Up:

1.1) Brief about version control :

Version control is used to keep track and maintain your versions of the code. In every iteration you keep an eye if it was that the word on the changes made to the code, but why? Simply we can say that for avoiding conflicts and to raise also the collaboration between the members that are currently working on the project also to compare work and been more productive. And lastly to decide by the leader either to accept changes or not and also for making backups of the version of the codes in case something happened to the computer you working from or the OS just crashed.

2) More of bash:

2.1) Brief :

This class you will learn and been able to manipulate more of the intermediate bash commands and do complicated operations and changes in your current OS.

2.2) Starting with table of commands :

Command name	Short Description of it's job	Common flags
chmod	Changes the permissions of file/folder	+(permission) -(permission) -R
cut	For cutting peace of text in your terminal	-f -d
touch	Mainly used for creating files	None
grep	Used to search for words in given text	-i -v
cat	For displaying contents of a file	-n

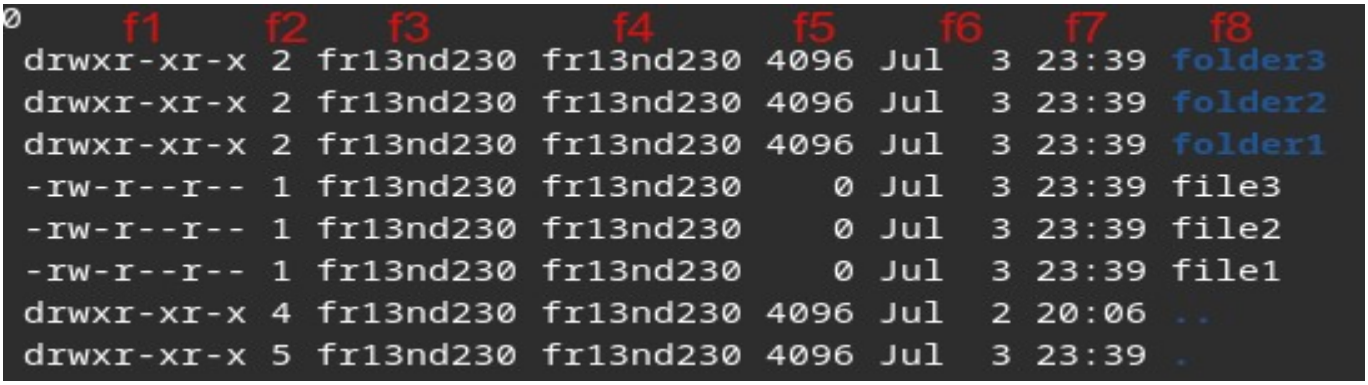
2.2.1) More about "chmod" command :

In linux systems every file should have an according permissions, those will help us maintaining our file (imagine you have a file named "password.key" and you are willing that you are just the one who wants to read and write to that file) so "chmod" or changing permissions of that file will help.

Permissions in linux systems are three (read,write and execute) or the triple of rwx, every one of them can have only two values [0 and 1], exactly! It's binary, r=0 means that you can't read that file in the other hand r=1 means that absolutely you can read that file! But how can we remember binaries in fact we don't we simply can remember which permission and what's in decimal (permissions starts from 0 until 7) as the following table:

Read (r)	Write (w)	Execute (x)	Decimal	Description
0	0	0	0	No permissions at all
0	0	1	1	Execute only
0	1	0	2	Write only
0	1	1	3	Write and execute
1	0	0	4	Read only
1	0	1	5	Read and execute
1	1	0	6	Read and write
1	1	1	7	Full permissions

Now how can we see the permissions or files or folders, simply we run the command "ls -aril" so we have 8 fields shown as following:



as you see we listed all information about those files and folders now let's focus on those fields.

Basically f1 is the permissions field is something like -|---|---|--- or if there was an text d/-|rwx|rwx|rwx notice that we separated them, the first where there is "d" and sometimes "-" specifies the type (d stands for directory , - means a file) then first set of permissions we call them owner's and he is you and me in that case the original person who created that file second set will be the group's which mean every person in same group as you will have same permissions and third set for the world and means the all other users in same system.

Now how could we change permissions, remember that table? Will help us a lot. Now there is many ways to change them with chmod command but we will talk on the efficient way first then we see the other methods:

First method: chmod <OWNER|GROUP|WORLD> file/folder

Note that each one of <OWNER|GROUP|WORLD> should be a permission in decimal like we want “file1 to readable and writeable for owner and world for group none we should use:

< user@PC:~$ chmod 606 file1 >

and so one for folders as well.

Second method : ugoa method for references

Now we can do quick adding directly permissions or subtracting permissions from a file +perm/-perm (example: +r make it readable | -w make it not writeable) we shall use:

< user@PC:~$ chmod [u|g|a] +/- perm >

< user@PC:~$ chmod a -x file1 >

now the “uga” means [u: for owner (user) | g: for group | a: for all users] and [+/- x | w | r].

This way is quick when we want to change a file permission for one set like owner and just one or two perm so no need for the decimal way.

This should wrap it up all you need to know about chmod is here in this quick resume.

2.2.2) More about “grep” & “cut” command :

Now the grep command and like it’s other sisters commands like “awk” or “sid” used for searching in text and outputting the result of the text, it returns the or all sentences which that word or symbol you searching for in and it’s one of the skills that every one should know about and it’s essential for linux users to get to know grep and for her sisters it’s for advanced classes. Now grep use piping “|” that symbol literally which means run the second command based of what first has gave as following < user@PC:~& echo “hello world” | tee -a file > which means echo that sentence and create a file which will be ‘hello world’ appended to that’s what “-a” in “tee” means. Let’s say we have dates in “dates.txt” as following:

Tue 5th july 2002 , Tue 6th july 2003 , Tue 10th june 2004 , Tue 5th july 2002 , Tue 4th jan 2021 , Tue 5th july 2002 Tue 5th july 2002

Notice that some dates are repeated we will benefit that now we want to grep "get" the all texts that have 5 or th in them simply we do: < user@PC:~$ cat dates.txt | grep 5th > or you can specify the word you are searching for between " " if you encountered a case which you word contains a symbol as '-'.

```
fr13nd230@FR13ND:~/Desktop/teaching/session2$ cat dates.txt | grep 5th
Tue 5th july 2002 , Tue 6th july 2003 , Tue 10th june 2004 , Tue 5th july 2002 , Tue 4th jan 2021 , Tue 5th july 2002 Tue 5th july 2002
```

as you see the result was all the text containing 5 or th in it. Let's say we don't want to search for the text which contains hello in "greeting.txt" , we add "-v" flag as:

< user@PC:~$ cat greeting.txt | grep -v hello >

```
fr13nd230@FR13ND:~/Desktop/teaching/session2$ cat greeting.txt ; #Simply we are ouputing the file contents
hello world english!
hello familly english!
hello guys english!
nihao ma chinese!
hola spanich!
Salam allaikum arabic!
hola amigos spanich!
hello friends spanich!
fr13nd230@FR13ND:~/Desktop/teaching/session2$ cat greeting.txt | grep -v hello
nihao ma chinese!
hola spanich!
Salam allaikum arabic!
hola amigos spanich!
```

now let's say that we wanted that text which doesn't contains "hello" in it but also we want to see what's language was said in and yes as you guessed we pipe the previous commands with "cut" command which has two flags we need: ["-d": delimiter used to tell where to cut the sentence and must be a single character | "-f": field in numbers to tell which word like: cut -d "ma" -f 1 will give us blank because first field after "ma" is space -f 2 will give "chinese!"]. So we shall search for text that doesn't contain hello and what comes after the space.

< user@PC:~$ cat greeting.txt | grep -v hello | cut -d "" -f 0 >

```
fr13nd230@FR13ND:~/Desktop/teaching/session2$ cat greeting.txt ; #Simply we are ouputing the file contents
hello world english!
hello familly english!
hello guys english!
nihao ma chinese!
hola spanich!
Salam allaikum arabic!
hola amigos spanich!
hello friends spanich!
fr13nd230@FR13ND:~/Desktop/teaching/session2$ cat greeting.txt | grep -v hello | cut -d ' ' -f 3
chinese!

arabic!
spanich!
```

Now note that “hola spanich!” returned nothing because there is no third “3rd” fields if we used -f 2 we should see “spanich!” but also we should see “amigos allaikum and friends”.

This should wraps all about “grep” and “cut” and searching for instances.

3) Get started with own git repositories:

3.1) Keep in mind :

Git is the tools that facilitates to us the process of maintaining our codes where GitHub is the cloud where we can share the final version or at least what we achieved and contribute to others codes.

3.2) More of git commands :

3.2.1) Initialize a repository :

Now for you to start with git first we must clarify that repository=folder and as developer your code files are in a fodler so how to make it a repository or better saying initialize it? Easy simple question that has an simple answer. Navigate to you folder and do:

< user@PC:~/Desktop/work/code$ git init . >

which means initialize and empty git repository in here.

Congratulations you have your first repository.

3.2.2) Watch for changes :

As you learned in coding and bash you will perform CRUD operations on your working folder like updating code , creating script or even worse getting mad and deleting your script and our job is to add and watch for this changes. How do we know what changes are made simply we use the ‘status’ option wich will tell us what files are added and ready for acceptance and which are not those we will add them with the add options as following:

< user@PC:~/Desktop/work/code$ git status >

will tell us which files are not added and those will be in red (red scary) if nothing means you accepted all and if green mean that you are waiting to accept. Or you can use “-s” for short

status as:

< user@PC:~/Desktop/work/code$ git status -s >

```
X          Y     Meaning
-------------------------------------------------
           [AMD]   not updated
M          [ MTD]  updated in index
T          [ MTD]  type changed in index
A          [ MTD]  added to index
D                  deleted from index
R          [ MTD]  renamed in index
C          [ MTD]  copied in index
```

And you can add the changes either by “add filename” or “add .” which means all as:

< user@PC:~/Desktop/work/code$ git status -s >

And now is you use “git status” you should see all in green.

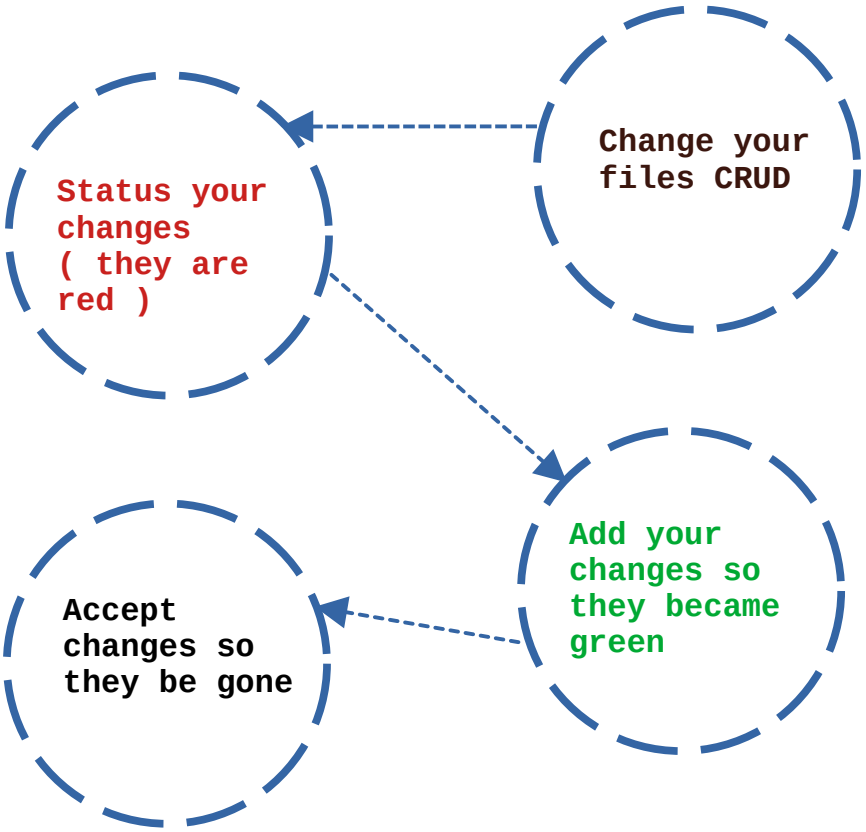
< Small break with a meme: >

Which will lead us what are commits?

3.2.3) Accept changes :

That acceptance your waiting for is what we call committing changes and that last need just to specify that yes I do accept all changes occurred to repository and ready for it and that can done simply by the option “commit -m <your message>” that message must be meaningful not like “Birdie Birdie I watched it yellow on morning” should be more like “Commit after fixing README.md” or “Fixed the issues of callback.js” and you can run it after seeing the status and adding all changes for committing as:
< user@PC:~/Desktop/work/code$ git commit -m “First Commit” >

Please consider and keep in mind that work flow :



See you in next class