Linux Fundamentals 01

echo : output any text that we provide

ex : echo "Hello World!"

whoami : find out what user is currently logged in

ex : whoami

ls : listing

first navigate to the file or directory, then type this command to see what exists in the file or directory

ex : ls

a. ls <file or directory name> : using this command we do not need to go inside of a particular directory or file.

directly we can see what is the content inside in the directory or file.

ex : ls Pictures

cd : change directory

this is used to change current working directory.

ex : cd Pictures

cat : concatenate

used to vizualize the content of a particular file.(not just text files.!)

ex : cat todo.txt

a. cat <path to file> : in this case you can vizualize the content of the file without going to the file location.

ex : cat /home/ubuntu/Document/todo.txt

pwd : print working directory

ex: pwd

find : we already know the name of the file we're looking for — but can't remember where it is exactly!. In this case, we're looking for "passwords.txt".

If we remember the filename, we can simply use,

ex : find -name passwords.txt

where the command will look through every folder in our current directory for that specific file

a. Find by the extension of a file,

But let's say that we don't know the name of the file, or want to search for every file that has an extension such as ".txt". Find let's us do that too!

We can simply use what's known as a wildcard (*) to search for anything that has .txt at the end. In our case, we want to find every .txt file that's in our current directory.

We will construct a command such as,

ex : find -name *.txt

Where "Find" has been able to find every .txt file and has then given us the location of each one.

grep : Grep can be used to look for lines of text that match one or more regular expressions, and it only displays the lines that match by default.

ex : grep "81.143.211.90" access.log

this command search all the records in the access.log file to find out this ip address and if found, then retrieve that record.

wc : The wc (word count) command is used in Linux systems to count the number of words, lines, and bytes in a text file.

ex : wc -l access.log

this gives us how many records inside the access.log file

shell operators

01. Operator "&"

This operator allows us to execute commands in the background.

For example, let's say we want to copy a large file. This will obviously take quite a long time and will leave us unable to do anything else until the file successfully copies.

The "&" shell operator allows us to execute a command and have it run in the background (such as this file copy) allowing us to do other things!

02. Operator "&&"

This shell operator is a bit misleading in the sense of how familiar is to its partner "&".

Unlike the "&" operator, we can use "&&" to make a list of commands to run for example command1 && command2.

However, it's worth noting that command2 will only run if command1 was successful.

03. Operator ">"

This operator is what's known as an output redirector. What this essentially means is that we take the output from a command we run and send that output to somewhere else.

running something such as echo howdy will return "howdy" back to our terminal — that isn't super useful. What we can do instead, is redirect "howdy" to something such as a new file!

ex : Let's say we wanted to create a file named "welcome" with the message "hey". We can run echo hey > welcome where we want the file created with the contents "hey".

Note: If the file "welcome" already exists, the contents will be overwritten!

04. Operator ">>"

This operator is also an output redirector like in the previous operator (>) we discussed.

However, what makes this operator different is that rather than overwriting any contents within a file, for example, it instead just puts the output at the end.

Following on with our previous example where we have the file "welcome" that has the contents of "hey". If were to use echo to add "hello" to the file using the > operator, the file will now only have "hello" and not "hey".

The >> operator allows to append the output to the bottom of the file — rather than replacing the contents.

Ex : echo hello >> welcome then welcome file include both, hey and welcome.