# Linux Fundamentals Part 1

## Learned:

- Being able to understand the file system by using commands such as ls, cd, or cat can allow you to move around directories and can help identify suspicious files or potentially malware.

- Using Shell Operators is a way to make running multiple commands easier, for example running the `&` operator while using the `cd` command can allow you to change directories in the background while you run other tasks.

- The Grep command can allow you to track sites an IP Address visited recently on a web server. Knowing what the user did is important in protecting sites.

## Flavor's of Linux

The name "Linux" is actually an umbrella term for multiple OS's that are based on UNIX (another operating system). Thanks to Linux being open-source, variants of Linux come in all shapes and sizes - suited best for what the system is being used for.

For example, Ubuntu & Debian are some of the more commonplace distributions of Linux because it is so extensible. I.e. you can run Ubuntu as a server (such as websites & web applications) or as a fully-fledged desktop.

# Commands

This is what a terminal looks like

```
tryhackme@linux1:~$ enter commands here
```

- You can do basic functions like navigate to files, output their contents and make files.

### Two first commands

| | |
|---|---|
| Command | Description |
| echo | Output any text that we provide |
| whoami | Find out what user we're currently logged in as |

Below is an example of each command being used

### Using echo

```
tryhackme@linux1:~$ echo Hello
Hello
tryhackme@linux1:~$ echo "Hello Friend!"
Hello Friend!
```

As shown in the terminal above, if we want to "echo" a single word, we don't need to use double quotes, for example, `echo Hello`. However, the string should be enclosed within double quotes if one or more spaces are present, for example, `echo "Hello Friend!"`.

`whoami` can be used to find the username we are logged in as.

Using whoami to find out the username of who we're logged in as

```
tryhackme@linux1:~$ whoami
```

## Interacting With the Filesystem

| | |
|---|---|
| Command | Full Name |
| ls | listing |
| cd | change directory |
| cat | concatenate |
| pwd | print working directory |

### Listing Files (ls)

Using "ls" to list the contents of the current directory

```
tryhackme@linux1:~$ ls
'Important Files' 'My Documents' Notes Pictures
```

In the screenshot above, we can see there are the following directories/folders:

- Important Files
- My Documents
- Notes
- Pictures

- *You can list the contents of a directory without having to navigate to it by using ls and the name of the directory. I.e.* `ls Pictures`

### Changing Our Current Directory (cd)

Now that we know what folders exist, we need to use the "**cd**" command (short for **c**hange **d**irectory) to change to that directory. Say if I wanted to open the "Pictures" directory - I'd do "**cd Pictures**". Where again, we want to find out the contents of this "Pictures" directory and to do so, we'd use "**ls**" again:

Listing our new directory after we have used "cd"

```
tryhackme@linux1:~/Pictures$ ls
dog_picture1.jpg dog_picture2.jpg dog_picture3.jpg dog_picture4.jpg
```

In this case, it looks like there are 4 pictures of dogs!

### Outputting the Contents of a File (cat)

"Cat" is short for concatenating & is a way for view  contents of files.

In the screenshot below, you can see how I have combined the use of "ls" to list the files within a directory called "Documents":

### Using "ls" to list the contents of the current directory

```
tryhackme@linux1:~/Documents$ ls
todo.txt
tryhackme@linux1:~/Documents$ cat todo.txt
Here's something important for me to do later!
```

- You can use cat to output the contents of a file within directories without having to navigate to it by using cat and the name of the directory. I.e*.* `cat /home/ubuntu/Documents/todo.txt`

- Sometimes things like usernames, passwords, flags or configuration settings are stored within files where "cat" can be used to retrieve these.

### Finding out the full Path to our Current Working Directory (pwd)

You'll notice as you progress through navigating your Linux machine, the name of the directory that you are currently working in will be listed in your terminal.

It's easy to lose track of where we are on the filesystem exactly, which is why I want to introduce "**pwd**". This stands for **p**rint **w**orking **d**irectory.

Using the example machine from before, we are currently in the "Documents" folder — but where is this exactly on the Linux machine's filesystem? We can find this out using this "pwd" command like within the screenshot below:

### Using "pwd" shows the full path of the current folder.

```
tryhackme@linux1:~/Documents$ pwd
/home/ubuntu/Documents
tryhackme@linux1:~/Documents$
```

## Searching for Files

Theirs no need to use `cd` and `ls` to find out what is where. Instead, you can use commands such as `find` .

## Using Find

If we remember the filename, we can simply use `find -name passwords.txt` where the command will look through every folder in our current directory for that specific file like so:

Using "find" to find a file with the name of "passwords.txt"

```
tryhackme@linux1:~$ find -name passwords.txt
./folder1/passwords.txt
tryhackme@linux1:~$
```

"Find" has managed to *find* the file — it turns out it is located in folder1/passwords.txt

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

Using "find" to find any file with the extension of ".txt"

```
tryhackme@linux1:~$ find -name *.txt
./folder1/passwords.txt
./Documents/todo.txt
tryhackme@linux1:~$
```

Find has managed to *find*:

1. "passwords.txt" located within "folder1"
2. "todo.txt" located within "Documents"

## Using Grep

The `grep` command allows you to search the contents of files for specific values that we are looking for. For example, the access log of a web server. In this case, the access.log of a web server has 244 entries.

### Using "wc" to count the number of entries in "access.log"

```
tryhackme@linux1:~$ wc -l access.log
244 access.log
tryhackme@linux1:~$
```

Let's say if you wanted to search this log file to see the things that a certain user/IP address visited? Looking through 244 entries isn't efficient considering we want to find a specific value.

We can use `grep` to search the entire contents of this file for any entries of the value that we are searching for. Going with the example of a web server's access log, we want to see everything that the IP address "81.143.211.90" has visited.

### Using "grep" to find any entries with the IP address of "81.143.211.90" in "access.log"

```
tryhackme@linux1:~$ grep "81.143.211.90" access.log
81.143.211.90 - - [25/Mar/2021:11:17 + 0000] "GET / HTTP/1.1" 200 417 "-"
"Mozilla/5.0 (Linux; Android 7.0; Moto G(4))"
tryhackme@linux1:~$
```

"Grep" has searched through this file and has shown us any entries of what we've provided and that is contained within this log file for the IP.

## Shell Operators

| Symbol / Operator | Description |
|---|---|
| & | This operator allows you to run commands in the background of your terminal. |
| && | This operator allows you to combine multiple commands together in one line of your terminal. |
| > | This operator is a redirector - meaning that we can take the output from a command (such as using cat to output a file) and direct it elsewhere. |
| >> | This operator works like the `>` operator, but instead of replacing the output, it adds (appends) it to the existing content. |

## Operator "&"

This operator allows us to execute commands in the background. For example, if you want to copy a large file.

The "&" shell operator allows you to execute a command and have it run in the background.

## Operator "&&"

Unlike the **"&"** operator, we can use "&&" to make a list of commands to run for example `command1 && command2`. However, `command2` will only run if `command1` was successful.

## Operator ">"

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

An example of this is redirecting the output of the `echo` command.

Running something such as `echo howdy` will return "howdy" back to the terminal but, What you can do instead, is redirect "howdy" to something such as a new file.

Let's say you wanted to create a file named "welcome" with the message "hey". you would run the command `echo hey > welcome` where we want the file created with the contents "hey".

### Using the > Operator

```
tryhackme@linux1:~$ echo hey > welcome
```

#### Using cat to output the "welcome" file

```
tryhackme@linux1:~$ cat welcome
hey
```

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

## 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.

## Using the >> Operator

```
tryhackme@linux1:~$ echo hello >> welcome
```

Using cat to output the "welcome" file

```
tryhackme@linux1:~$ cat welcome
hey
hello
```

If you wanted to replace the contents of a file named **"passwords"** with the word **"password123"**

**echo password123 > passwords**

Now if I wanted to add "tryhackme" to this file named **"passwords"** but also keep **"passwords123"**

**echo tryhackme >> passwords**