**Linux for beginners**

**structure:**

Part 1 : We will look at 3 modules from tryhackme(linux fundamentals 1,2,3) where we will look at beginner commands.

Part 2 : We will cover Linux for hackers book(11 chapters)

Estimated time : 3 + 11 = total 14 working days

Tryhackme : 3 modules YouTube links below. In case of any doubts regarding the 3 modules you can either watch the youtube videos or reach me.

<https://www.youtube.com/watch?v=kPylihJRG70&ab_channel=TryHackMe>

<https://www.youtube.com/watch?v=7Zt2Mp2IeBI&ab_channel=TryHackMe>

<https://www.youtube.com/watch?v=bwgaZCb2ft8&ab_channel=TryHackMe>

**Linux fundamentals 1**

You're most likely using a Windows or Mac machine, both are different in visual design and how they operate. Just like Windows, iOS and MacOS, Linux is just another operating system and one of the most popular in the world powering smart cars, android devices, supercomputers, home appliances, enterprise servers, and more.

**Where is Linux Used?**

It's fair to say that Linux is a lot more intimidating to approach than Operating System's (OSs) such as Windows. Both variants have their own advantages and disadvantages. For example, Linux is considerably much more lightweight and you'd be surprised to know that there's a good chance you've used Linux in some form or another every day! Linux powers things such as:

* Websites that you visit
* Car entertainment/control panels
* Point of Sale (PoS) systems such as checkout tills and registers in shops
* Critical infrastructures such as traffic light controllers or industrial sensors

## Flavours of Linux

The name "Linux" is actually an umbrella term for multiple OS's that are based on UNIX (another operating system). Thanks to UNIX being open-source, variants of Linux comes 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.

Similar to how you have different versions Windows (7, 8 and 10), there are many different versions/distributions of Linux.

As we previously discussed, a large selling point of using OSs such as Ubuntu is how lightweight they can be. This, of course, doesn't come without its disadvantages, where for example, often there is no GUI (Graphical User Interface) or what is also known as a desktop environment that we can use to interact with the machine (unless it has been installed). A large part of interacting with these systems is using the "Terminal".

***\*\*\*\*Discuss cmd help and sudo(#) and normal user($)***

The "Terminal" is purely text-based and is intimidating at first. However, if we break down some of the commands, after some time, you quickly become familiar with using the terminal!

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

Graphical user interface, text, application

Description automatically generated

## Interacting With the Filesystem

As I previously stated, being able to navigate the machine that you are logged into without relying on a desktop environment is pretty important. After all, what's the point of logging in if we can't go anywhere?

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

### **Listing Files in Our Current Directory (ls)**

Before we can do anything such as finding out the contents of any files or folders, we need to know what exists in the first place. This can be done using the "ls" command (short for listing)

### **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:

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

Whilst knowing about the existence of files is great — it's not all that useful unless we're able to view the contents of them.

We will come on to discuss some of the tools available to us that allows us to transfer files from one machine to another in a later room. But for now, we're going to talk about simply seeing the contents of text files using a command called "**cat".**

"Cat" is short for concatenating & is a fantastic way us to output the contents of files (not just text 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":

*cat /home/ubuntu/Documents/todo.txt*

### **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 Find****

The find command is fantastic in the sense that it can be used both very simply or rather complex depending upon what it is you want to do exactly

Now, of course, directories can contain even more directories within themselves. It becomes a headache when we're having to look through every single one just to try and look for specific files. We can use find to do just this for us!

Let's start simple and assume that 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 find -name passwords.txt where the command will look through every folder in our current directory for that specific file like so:

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

Another great utility that is a great one to learn about is the use of grep. The grep command allows us to search the contents of files for specific values that we are looking for.

Wc – word count

wc -l file\_name

$wc -l access.log

Using a command like cat isn't going to cut it too well here. Let's say for example if we wanted to search this log file to see the things that a certain user/IP address visited? Looking through 244 entries isn't all that 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 (note that this is fictional)

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))"

Linux operators are a fantastic way to power up your knowledge of working with Linux. There are a few important operators that are worth noting. We'll cover the basics and break them down accordingly to bite-sized chunks.

At an overview, I'm going to be showcasing the following 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 does the same function of the > operator but appends the output rather than replacing (meaning nothing is overwritten). |

Let's cover these in a bit more detail.

## 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!

## 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.

## 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.

A great example of this is redirecting the output of the echo command that we learned in Task 4. Of course, 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!

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" like so:

Using the > Operator

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

Using cat to output the "welcome" file

tryhackme@linux1:~**$** cat welcome

hey

*Note: If the file i.e. "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 like so:

Using the >> Operator

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

Using cat to output the "welcome" file

tryhackme@linux1:~**$** cat welcome

hey

hello