

LINUX FOR BEGINNERS

By Syed Abdullah Shah

Introduction

Sunday, 21 July 2024 12:49 PM

Welcome to my rendition of Linux for beginners.

My name is Syed Abdullah Shah and I'll be your guide within the pages of these notes, I am in my junior year of a bachelors in Computer Science and well, I have a keen interest in Cybersecurity. If you have any questions about any information within these notes, feel free to hit me up on [LinkedIn](#), or via Email at aosawn@gmail.com.

These notes focus on the very basic foundations that will help you establish a grip on the Linux shell.

This learning path assumes that you have:

1. Intermediate level of computing knowledge
2. Knowledge about the basics of Operating systems
3. A passion to learn

Why linux?

In the world of computing, there exists a versatile and robust operating system that has been quietly shaping the digital landscape for decades. Linux, often referred to as the “Swiss Army knife” of operating systems, has grown from a humble open-source project to an integral part of our modern technological ecosystem. In this blog, we'll delve into the **Importance of Linux OS** and explore how it has revolutionized the way we use computers.

Greatest Features of Linux OS:

- **Open Source Philosophy**

At the core of Linux's significance lies its open-source philosophy. Unlike proprietary operating systems, Linux is developed collaboratively by a community of passionate programmers and developers from around the globe. This open nature encourages innovation, fosters transparency, and enables anyone to contribute and tailor the system to their needs. This results in a more secure, stable, and customizable environment, making Linux a preferred

choice for developers, researchers, and tech enthusiasts.

- **Customizability and Flexibility**

Linux OS provides an unparalleled level of customizability and flexibility. Users have the freedom to choose from a variety of “distributions” or “distros,” each tailored to specific use cases and preferences. Whether you’re a system administrator, a gamer, a creative professional, or a casual user, there’s a Linux distro designed just for you. The ability to customize everything from the user interface to the underlying system components empowers users to craft an environment that suits their unique needs and preferences.

- **Stability and Security**

A major advantage of the Linux operating system is considered to be its stability and security. Thanks to its modular architecture and robust design, Linux is known for its ability to run for extended periods without requiring a restart, making it ideal for critical applications and servers. Additionally, the open-source nature of Linux allows security vulnerabilities to be identified and addressed quickly by the global community, leading to faster patching and enhanced security compared to proprietary systems.

So, whether you are into hacking, cloud computing, or any other IT field, linux is important for you, and it's time you get your hands on it.

Let's get into it.

Virtual Machines & Setting up Linux

Sunday, 21 July 2024 1:14 PM

Now, you might have thought, how will I work and learn on linux, when I don't even have a linux system?

What if I told you that you can have a linux system, right here on the same windows computer that you use every day, right alongside the Windows OS. That's right, this can be achieved through the help of a fascinating thing called **Virtual Machines**.

So, what are virtual machines?

Virtual machines are a piece of software, that allows your computer to "virtually" run more than one operating system. Try to understand it like your computer is the "host" machine, and all the virtual machines it can run will be "guest" machines. Like every other application, virtual machines also take up resources from the host machine, resources like CPU core, RAM, Disk Storage etc. All while simultaneously running a totally different operating system than the host machine. It's basically a computer inside a computer. You can research more about virtual machines on your own, let's get to setting one up right now.

Setting up a virtual machine:

To set up a virtual machine, you need a virtualization software AKA a hypervisor. There are many out there, but the one we will be using is called **VirtualBox by Oracle**. Here's how to set it up.

Use the link below to download Virtualbox from the official website for free. The link below is for windows, but you can also download it for MacOS and other systems.

<https://download.virtualbox.org/virtualbox/7.0.20/VirtualBox-7.0.20-163906-Win.exe>

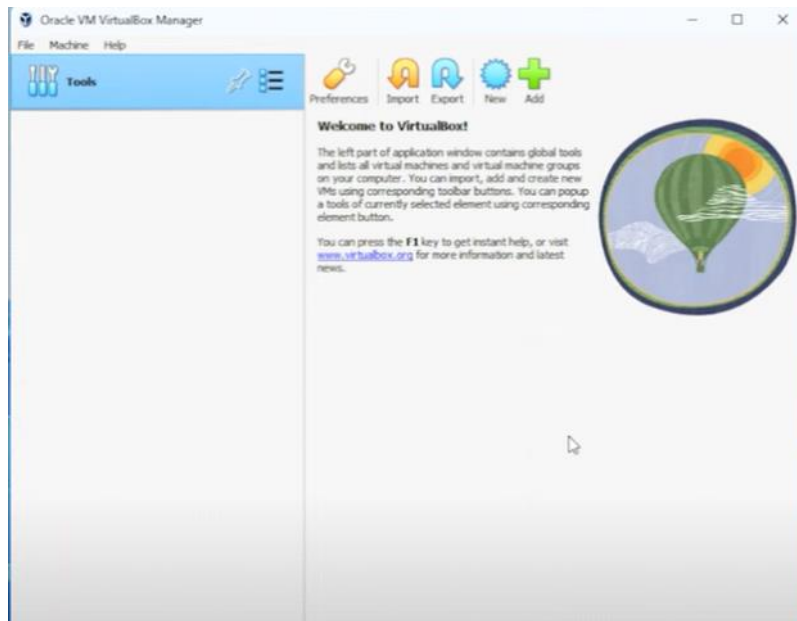
After completing the download, begin the installation process by clicking on the .exe setup that was installed.

Go through the installation process, no need to change any of the options during installation, default will always be the best way to go when you don't know about it. A warning will pop up during the installation about network interface, like below:



Just click yes and proceed through it.

After the installation is complete, run the Virtualbox, and you will end up with this window, yours might look a little different depending on the version of Virtualbox installed.



Once you're on this page, the next thing you need is an Operating system that you want to load into this hypervisor. Since we are going to be learning about Linux, we obviously need a distribution of Linux. If you're into hacking, you should download the Kali Linux distribution, it is a debian based distribution (Don't worry if you don't know what these terms mean just yet, you will know soon).

We will be importing Kali Linux into our hypervisor.

Downloading Kali Linux:

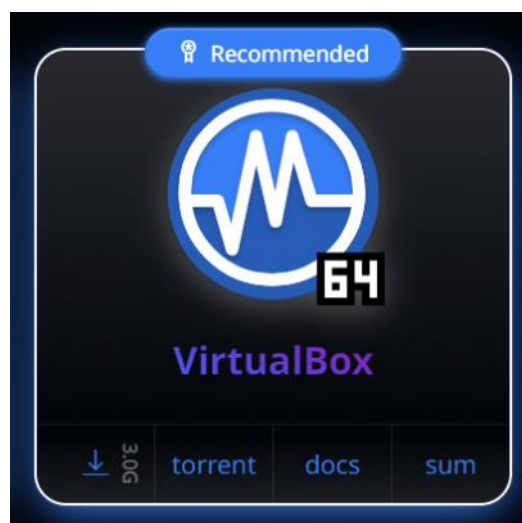
Navigate to the link below

<https://www.kali.org/get-kali/#kali-virtual-machines>

And there you will find various images for various hypervisors. What is an image you might ask?

An image is a pre-configured, fully installed version of an operating system, that is super quick to use and set up on a hypervisor. This saves us time while setting up a complete system for us.

Select the image for the Virtualbox hypervisor from the website, it'll look like this:



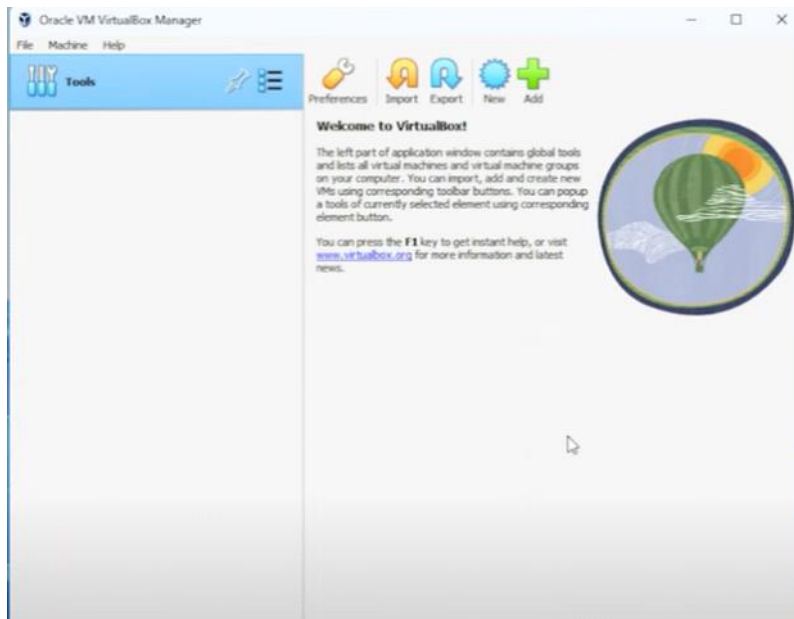
Download this file and keep in mind, it will take some time, as you are downloading a pre-configured operating system.

Setting up Kali Linux in VirtualBox:

Now, we will be setting up Kali linux in our virtual box.

Extract the the file that you downloaded from the Kali Linux website, and you will see a **.ISO** file in it.

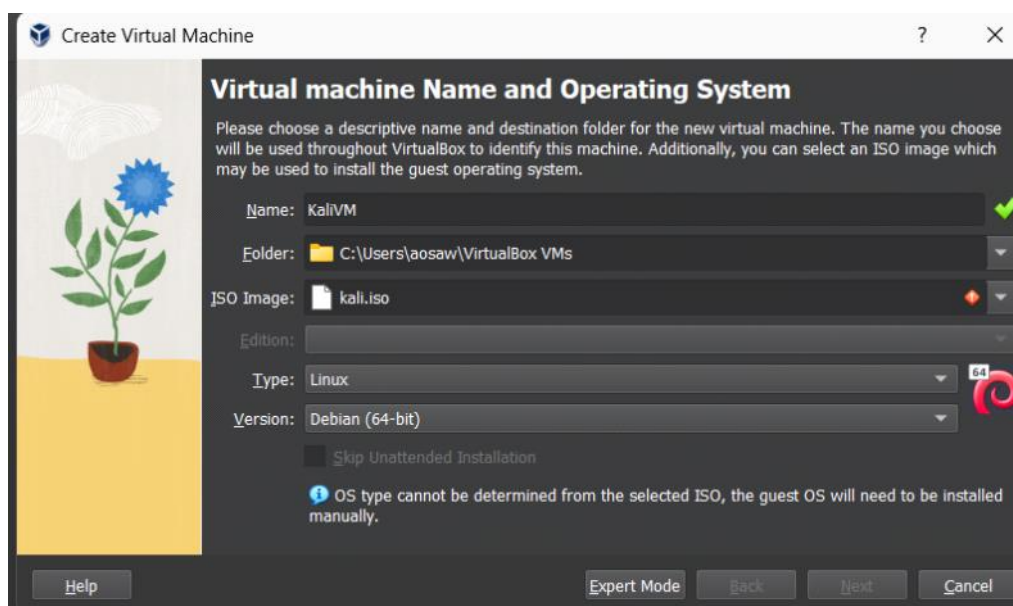
Now open Virtualbox, and follow the steps below.



Click on the "NEW" icon from the menu on the top right.

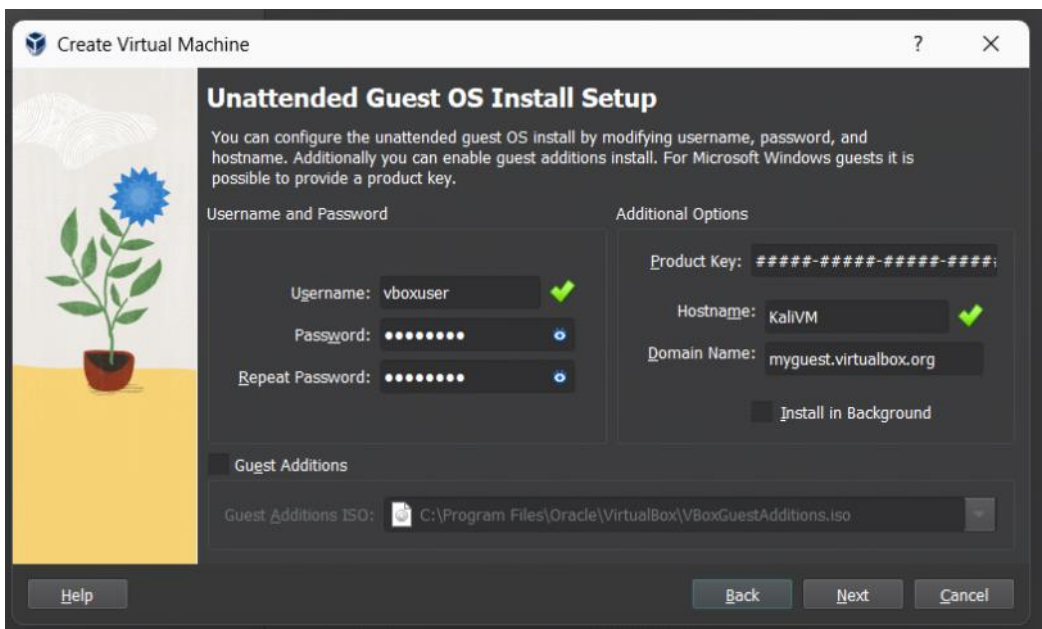
A dialogue box will open.

Configure it as such:

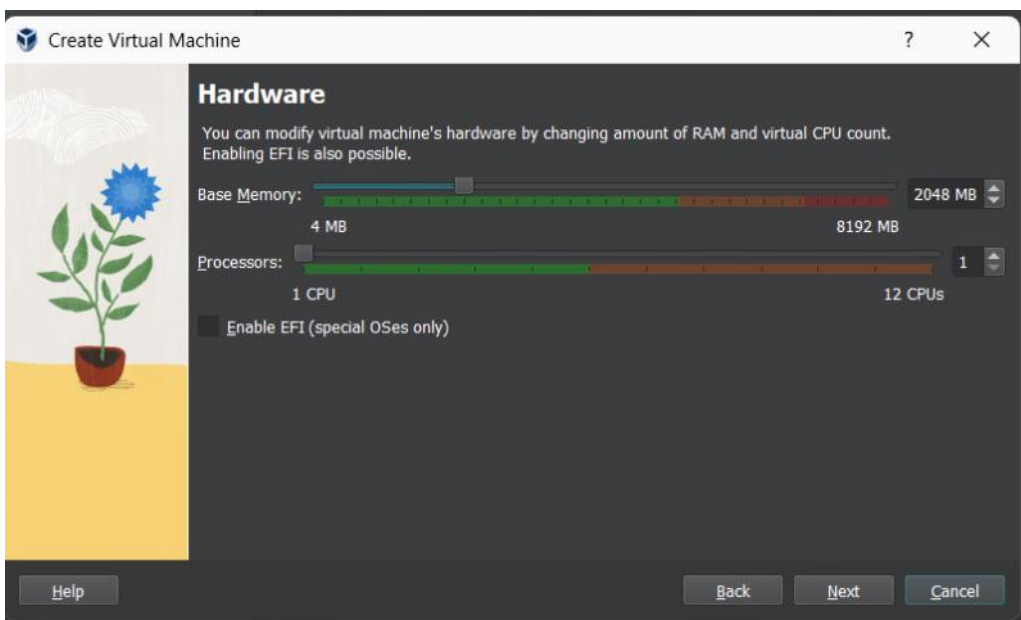


Load that iso file into this dialogue box.

After this, hit next.



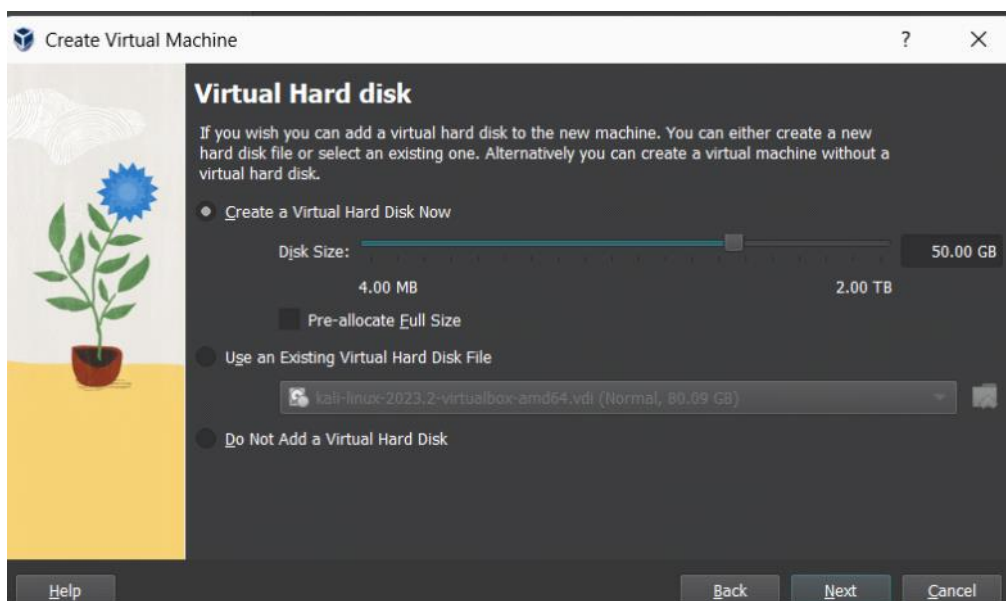
Hit next.



Now configure how much hardware resource you want to allocate to the virtual machine.

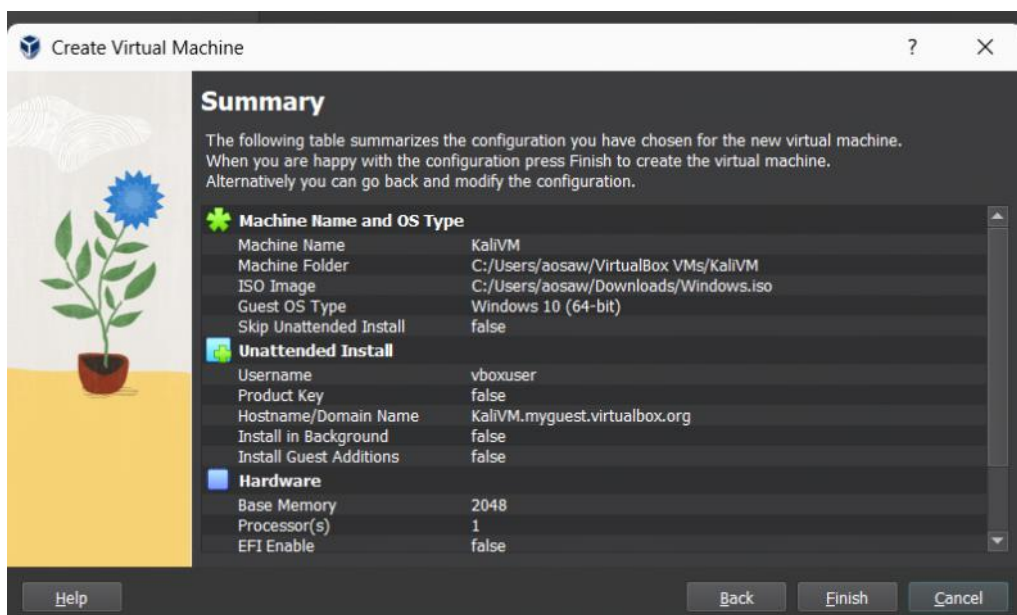
For a kali linux system, allocate atleast 2 GB of RAM, and 3 to 4 CPU's. Do not go more than half of your total resource as it will create problems for you in the future.

Hit next.



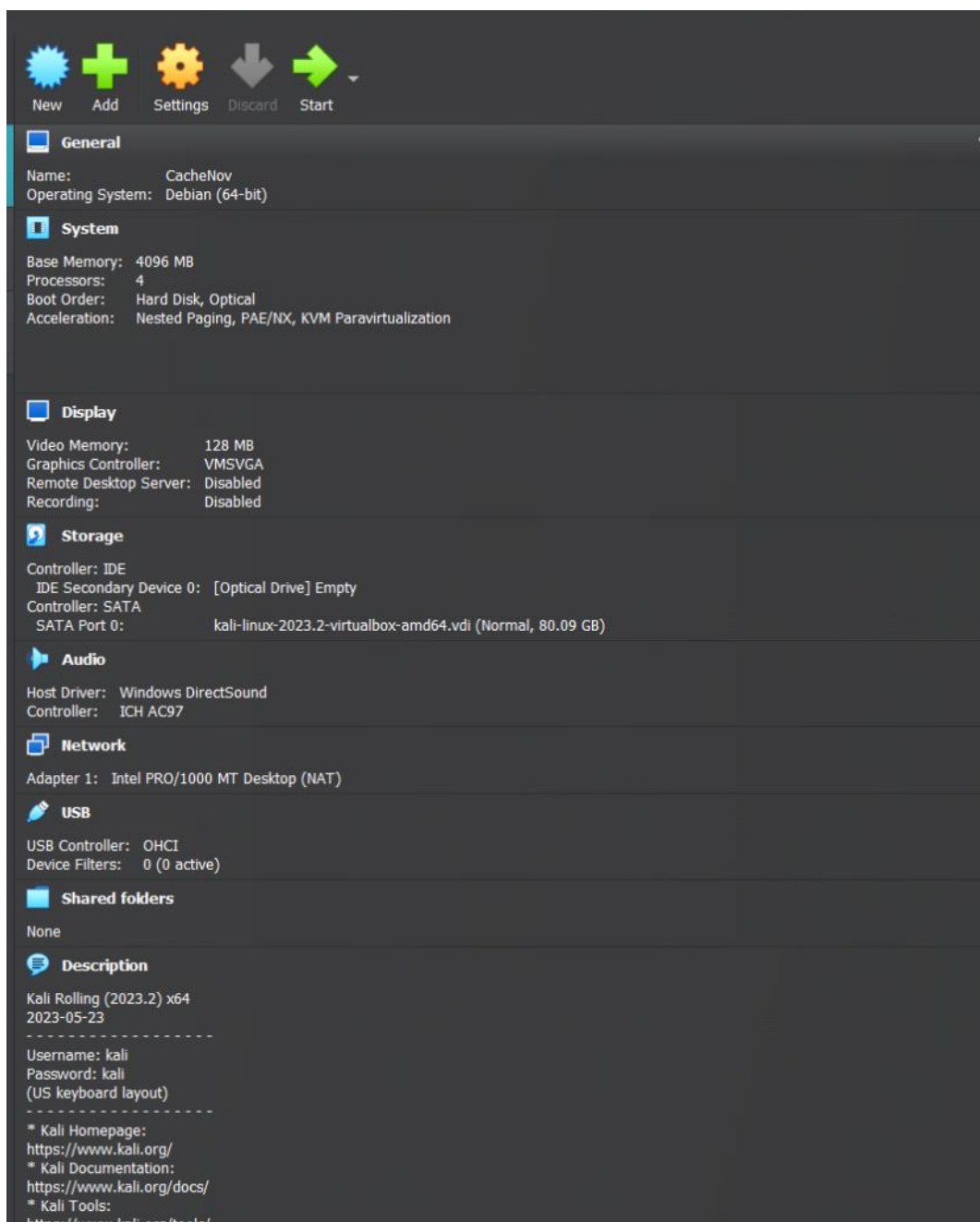
Allocate Hard disk space, and keep in mind that this will create a virtual hard disk partition on your main host hard drive. Allocate a minimum of 20 GB storage to this virtual hard drive. Do not check the "**PRE-ALLOCATE FULL SIZE**" as the entire mentioned storage will be hoarded by this virtual machine. By default it allocates dynamically, as much space is needed, within the limit of how much space you have assigned in the virtual hard drive.

Keep all settings as default and hit next.



Just review these settings and hit finish. The virtual machine will be set up.

After you're done, it should look something like this:

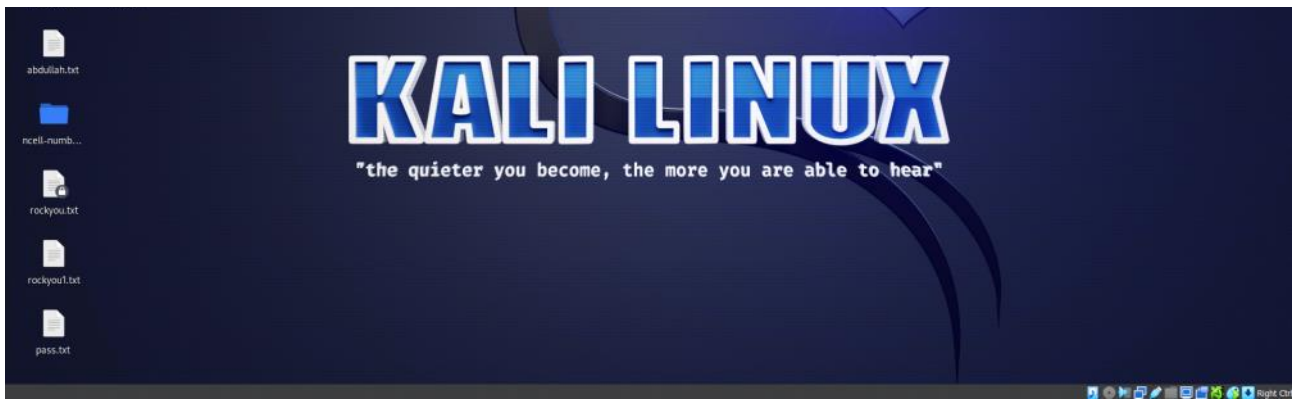


Now, press start to get into the system.

Your default username and password might be "kali" & "kali" or "vboxuser" & "changeme" respectively.

Congratulations, you have now launched Kali linux, a virtual machine, on your host machine.





You use your regular mouse and keyboard to interact with this system as well. Now in order to take back control and come to the host machine, just hit the RIGHT CTRL on your keyboard once, and you will see a green downward arrow turn from green to black, this means that everything you do on mouse and keyboard now will render on the host machine. Press it again to get back into the virtual machine.

That's all for this one, see you in the next one.

For queries, feel free to hit me up on [LinkedIn](#), or via Email at aosawn@gmail.com.

Basics

Wednesday, 17 July 2024 11:16 PM

Navigation:

Linux operates on a FHS based filing system, that means all files lead up to a root, in a tree-like hierarchy.

All that you can do in a GUI, you can do faster and more efficiently in a BASH/Terminal

Some of the basic file system navigation commands are as under:

1. To know where you are in terms of directory, you use the **PWD** command, also known as "Print working directory". This is similar to seeing the path displayed above any file explorer in a GUI
2. To see the files and folders present in any directory, we use "**LS**" command, that stands for LIST. This lists out all elements present within a directory.
3. To go into a subdir, we use the "**CD**" command, that stands for change directory. For example "cd Desktop" would take us from the home directory into the Desktop directory.
4. To go back from a directory, we use "**cd ..**" command, the two dots mean back.

Linux File System:

First command of the topic, to see who you are logged in as: use command "**whoami**", mostly used to see the current logged in user, especially helpful when taking remote shell connections.

The root of the linux file system, represented as **/**, this is the starting point of the linux file system, the beginning point of it all, all the subdirectories can be viewed and accessed from here. There will be some principles that we will be taking a look at, throughout these notes.

*Principle: **Everything in linux is a FILE.** Like EVERYTHING, configurations, network settings, device settings, all of them, stored somewhere, in single or multiple text files.*

To the point that, even all the commands used in linux, every single one of them, is a file.

This can be verified by navigating to the **bin** file and using the **ls** command to list its contents.

COMMAND:

To view, print contents of a file, you use the command "**cat**" that comes from concatenate.

So for example, viewing contents of the **ls** file in the **bin** folder, you write "**cat ls**". This prints the entire content of a file on the terminal bash.

COMMAND:

To run some special commands, we kind of require a special one time permission, or you can say we tell the system "**please**" run this command for me, this is done by using **SUDO** before any special command. Sudo stands for - Super user Do. It helps elevate permissions to root user level for that one specific command, without requiring to sign in as root completely.

COMMAND:

To make a copy of any file, we use the command "**cp**", standing for copy, so like syntax will be;

`Cp <filetocopy> <filetocopyinto>`

And by using this you can also make a copy of any command and rename it to whatever you like, kinda neat like making a shortcut of it

COMMAND:

To remove something or delete something, the command to be used is "**rm**", standing for remove, similar syntax to cp, just `rm <fileToBeDel>`.

NOTE: RM and CP require sudo permissions

Much like other commands, that can be found in the bin folder, there is another folder called **SBIN**, standing for super bin.

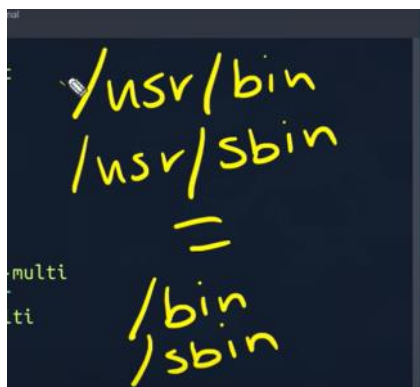
SBIN: This file contains all the commands that can only be used by administrators, to well, administer the system, and implement controls.

COMMAND:

To add a new user to the system, we use the command "adduser", this command can be found in the/sbin folder, since it is a special admin level command. By default, add user wants you to adhere to a regex, for easier searching but, you can just disable it temporarily by typing "--allow-bad-names" with the add user command. Syntax:

`sudo adduser --allow-bad-names <username>`

Okay so, now, dilemma time, if you ls root directory, you see that there is a **USR** directory, go into it and ls it, you will see that there are bin and/sbin here too, also containing MOST of the same commands as the root bin and/sbin, so, what is it? What's the use, what's the difference. The answer is, that, there is none, they are uhhh, samee, for the most part.



Now, the question rises that, when you run a command, WHICH one is being fetched, the one in root or the one in usr. You can check this by a command.

COMMAND:

You can check which command is being run from where, by using the command "**WHICH**", literally.

It tells you where the command file is being run from. It helps you in finding command binaries.

User created command binaries should and MUST be in the "LOCAL" directory present in the root folder.

SPEED ROUND:

```
[root@localhost ~]# ls
bin      home      lib32     media     root      srv      var
boot     initrd.img lib64      mnt       run       sys      vmlinuz
dev      initrd.img.old libx32     opt       sandbox  tmp      vmlinuz.old
etc      lib       lost+found proc      sbin     usr
```

ls the root directory, you see files, quick explanations below:

Boot: all the files your system needs to boot.

Var: Log files, web app files

Tmp: Files that are temporary, that go away after a reboot.

Lib: shared files that system uses for boot

Home: Where every user lives on the system, every user, except the root.

Root: This is where the root user lives, all by himself

Dev: Like we said before, EVERYTHING IS A FILE, this is where your devices, hardware lives as files. In this you can find your virtual drives, labelled as either VDA (VDA1) or SDA (SDA1), you can CAT them, too. (Ctrl C to exit the process)

Etc: Your system, server, network settings are stored in this file. (pronounced as EtSee file)

You can go to this dir, and see the network dir, go into it, and you will find an "INTERFACES" folder, you can cat that, and you will be able to see all your configured network interfaces and assigned IP addresses.

Media: If a USB is plugged into the system, it is **automatically** mounted onto your system into the media directory as a file.

Mnt: This also **mounts** drives, but this mounts the ones you mount manually. You use commands to mount drives.

That's all for this one, see you in the next one.

For queries, feel free to hit me up on [LinkedIn](#), or via Email at aosawn@gmail.com.

Help

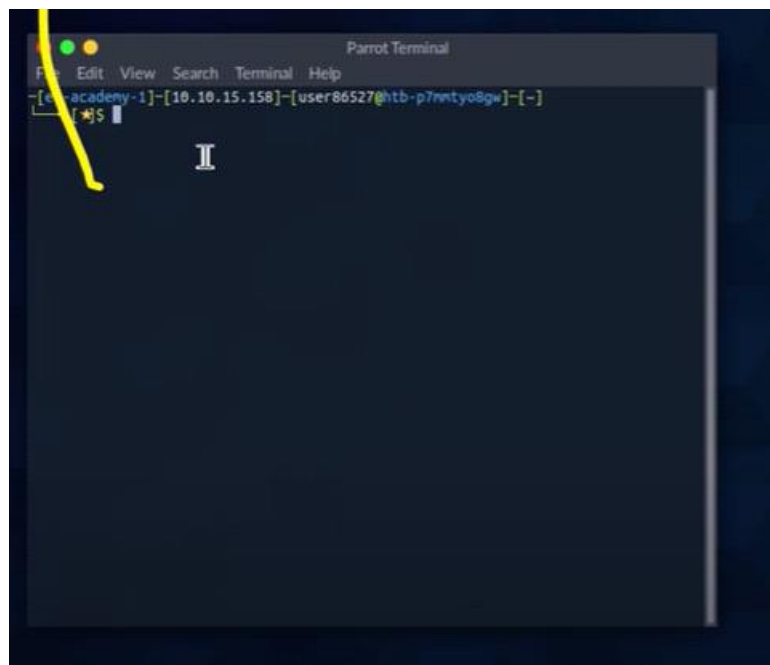
Wednesday, 17 July 2024 11:22 PM

You ever just want feel overwhelmed and stuck in the CLI, and you're like, I JUST NEED HELP.

Welp, Us bro us, but fear not, linux is there to help us. The beast will save us from the beast itself.

Okay, so, starting with the main element that we use, Mr. Terminal. What is a terminal?

To answer this question we first see that, which is NOT terminal.



This thing in the above image, IS NOT a terminal, rather it is a TERMINAL EMULATOR. So essentially, it emulates the terminal that was originally used to interact with any system. Below is an image of a real TERMINAL.



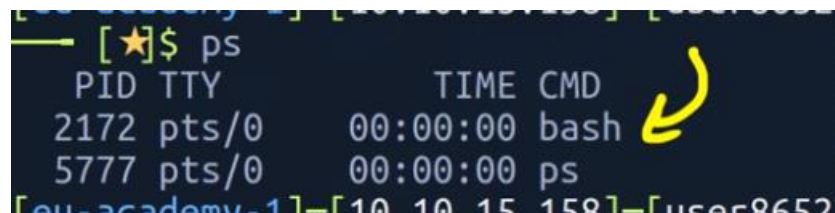
The terminal emulator has only one job - to give us access to "commands" to enable interaction with the system.

Okay so, the place where we TYPE the commands, that is NOT the terminal emulator, that is **THE SHELL**. Think of it like the terminal being the keyboard, and the CLI being the SHELL, thus Terminal enabling interaction with the shell. The most common shell used is called **BASH - Bourne again shell**.

Now, to check which shell you are running, or WHICH process you're running, you can use the following command.

COMMAND:

You can use the **PS** command, standing for Process status, this will display all current running processes on your system, one of them will show the type of shell running



```
[★]$ ps
  PID TTY          TIME CMD
 2172 pts/0        00:00:00 bash
 5777 pts/0        00:00:00 ps
```

You can also run powershell on linux. (powershell shows up as PWSH in process, it belongs to microsoft)

Okay, then, much like in cisco's IOS router config, the shell also has a SYMBOL denoting user levels.

As you can see in the above image, there is a \$ before the command "PS", that is a symbol.

\$ -> You are logged in as a USER

-> You are logged in as a ROOT user

HELP:

Now, linux can be overwhelming, especially with each command being able to be paired with other commands and tac flags(switches). It can be a bit too much to REMEMBER.

To get over this block, we use help commands.

COMMAND:

We use the "**man**" command paired with any command that we want to learn about, man stands for manual, it gives a detailed manual about any command you want. You can also use the "-h" switch to do essentially the same thing,

ORR you can use the "-help" switch to do it.

Now, a very fancy command, if you remember like what you want to do, but do not remember the command for it, you can figure it out through the command below.

COMMAND:

Use the command "**apropos**" along with a description of what you want to do, and it will find the command for you.

The 'apropos' command in Linux is used to search the man page descriptions for a specified keyword. It is used with the syntax, `apropos [keyword]` . It's like a search engine for your Linux commands, helping you find the right command for your needs.

That's all for now.

For queries, feel free to hit me up on [LinkedIn](#), or via Email at aosawn@gmail.com.

Managing Users

Friday, 19 July 2024 2:31 PM

Every computer system has - USERS.

Flashback: To quickly see which user you are, use the "whoami" command.

Command:

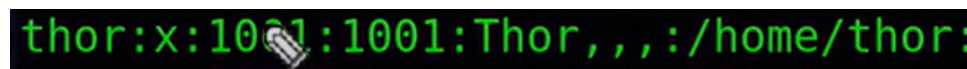
You can add a new user to the system, by using the "**adduser**" command.
(sudo command)

To Check where users are, ALL The users of the system,

Use -> **cat etc/passwd** , there you will find all the users present on the system, like accounts present.

Now, after you've catted the etc/passwd, you can see some information infront of every single user.

Details about that information:

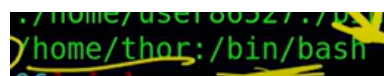


```
thor:x:1001:1001:Thor,,,:/home/thor:
```

The snapshot above shows that a user named "THOR" exists on the system. The "x" infront of it means that the password for this user is stored in some other file, namely a file called **etc/shadow**.

1001:1001 are his ID's, first one is UID and the second one is a GID (group ID) When a new user is created, a new group is also created, with that new user being it's only member right now.

After that it's random stuff and then his HOME directory, and default shell.



```
cat /etc/passwd  
thor:x:1001:1001:Thor,,,:/home/thor:/bin/bash
```

What does "useradd" do? What is the difference?

Uhmm, it essentially does the same thing, It creates a user, BUT, without any extra details, it doesn't assign a password, or other details which "adduser" asks for.

COMMAND:

To set a password for a user, you can use the command "**passwd**", along with sudo to assign a password to a user. It will show up in the shadow directory (a directory where password hashes are stored)

The downside to using "useradd" instead of "adduser" is that well the former is LAZY. It doesn't give a password, it doesn't assign a default bash, it doesn't assign a home directory for the user in the /usr directory.

By default, The user added by the useradd command will have the "sh" shell instead of BASH.

We can change that.

COMMAND:

You can modify a user account by using "usermod" command. You can use -> **usermod -h** to figure out what you can do with the switches. But just to change the shell, you can use:

sudo usermod <username> --shell /bin/bash

This will change the default shell of the given user to the shell BASH.

verify by cat etc/passwd

SUDO: Stands for "Super User Do", kinda like the infinity gauntlet of thanos, just for that specific command. Super user is the **root user**. But not everyone can use the sudo command.

COMMAND:

You can switch users by using the command "**su**", and then the username of who you want to login as.

Su - <username>

Note: if you use (SU -) without adding a username, you will be logged in as ROOT user.

```
testuser:x:1001:1001:linux prep,,,:/home/testu
ser:/bin/bash

(kali㉿kali)-[/]
$ su - testuser
Password:
(testuser㉿kali)-[~]
$ sudo adduser iman
[sudo] password for testuser:
testuser is not in the sudoers file.
```

If another user tries to use sudo, it will not be allowed, as it is not in "SUDOERS" file.

Sudoers file is a file which contains all the users that CAN use the sudo command.

Now, the fun part, as with every other file, THE SUDOERS FILE CAN BE CHANGED TOO!

(Don't do it, it'll mess up everything lol)

Okay so, the only best and safe way to edit sudoers file is to use the command -> **sudo visudo**

In there you can add a user in the user privilege specs list, this will allow the user to run sudo commands

<username> ALL = ALL

This means that the username will have access to ALL systems and ALL commands, that's what ALL=ALL stands for.

```
# User privilege specification
root    ALL=(ALL:ALL) ALL
thanos  ALL = ALL
# Allow members of group sudo to
```

Now, to delete any user

COMMAND:

We use the "userdel" command to delete any user, this is a sudo command, so sudo will be required.

Now, we explore GROUPS.

COMMAND:

To add a group, use the command "**groupadd**".

To see where the group was added, use -> **cat etc/group**

In there, you will be able to see the newly added group, along with groups for any new users you mightve added along the way. (remember each new user gets a new group too).

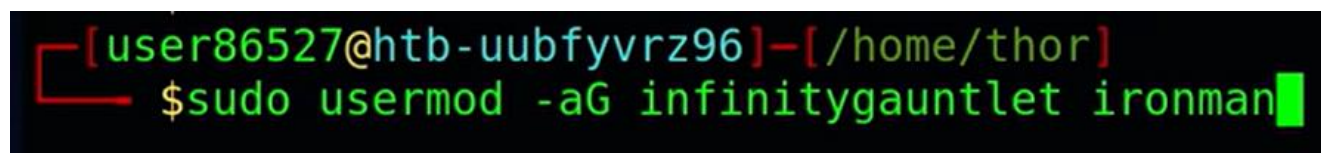
Now, to give a group the sudo powers, you add it into the sudoers file.

COMMAND: To check which groups YOU are a member of, use the "groups" command to see.

Now, to add a user to a group

COMMAND:

We use the **usermod** command with the switch **-aG** to add a user to the group, the -aG stands for APPEND GROUP. The user can be added to the group with just the -G switch, but what that does is it adds the user to the indicated group BUT it will remove it from every other group. -aG does it such that it appends the new group to existing groups.

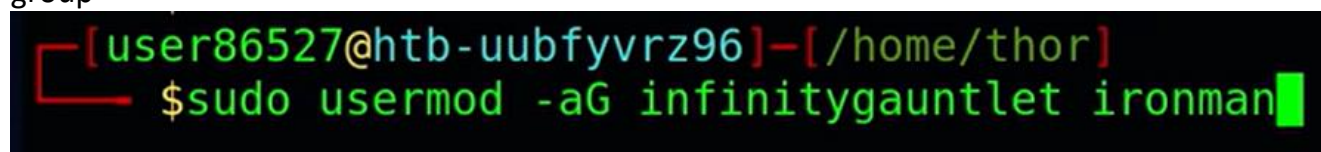
A terminal window with a black background. The prompt is '[user86527@htb-uubfyvrz96]-[/home/thor]'. The command '\$sudo usermod -aG infinitygauntlet ironman' is entered and executed, with a green cursor at the end.

Sudo usermod -aG <groupname> <username>

Now, to remove a user from the group

COMMAND:

We use the command **gpasswd** with the **-d** switch to remove a user from a group

A terminal window with a black background. The prompt is '[user86527@htb-uubfyvrz96]-[/home/thor]'. The command '\$sudo usermod -aG infinitygauntlet ironman' is entered and executed, with a green cursor at the end.

COMMAND:

To Delete a group, use the command **groupdel** to do it.

That's it about user management.

For queries, feel free to hit me up on [LinkedIn](#), or via Email at aosawn@gmail.com.

Linux Package Management

Friday, 19 July 2024 3:16 PM

How do you install anything on linux???

Anything you want to install, is contained in a package.
Thus, we install a package.

For that, we use package managers, there are two main ones.

1. dpkg
2. Apt

APT is super easy to use, thus we see dpkg first.

DPKG:

It is a low level package manager.

.DEB stands for debian , it is the extension of packages in debian based packages.

COMMAND:

Sudo Dpkg -i discord

Now, dpkg sucks!

WHY? Because well, you first have to download a package from the web, and then depackage it using dpkg, second, it has a lot of problems, like dependency issues. Like when you install a package, and it depends on other packages that should be present but aren't, it doesn't work.

APT: Apt stands for advanced package tool

It is a high level tool

COMMAND:

sudo apt install <packagename>

APT doesn't only install the package you specified, it will also install all the dependencies that are required by that package

APT relies on a repository, that has a collection of all the software that we need to install.

That is why it doesn't need the .deb package file pre installed.

GOOD PRACTICE: Always do **apt update** before doing **apt install**, thus allowing all the repos to be updated frequently.

COMMAND:

To see the repos that you have, you can use the command -> **sudo apt edit-sources**

COMMAND:

If you are printing an output, and you want to search for something specific within an output, use "GREP" command

So like for example, -> **apt list --installed | grep ^nmap**

This prints all the packages installed in our pc, and displays from them only the ones containing the keyword NMAP.

Now, how do we uninstall

COMMAND:

Use -> **sudo apt remove <packagename>** to remove any package from the system. Now, keep in mind that remove is a safe command, so like it only removes the app, but not the user data of the app, so any configurations or save files you may have made for the app, they won't be removed. Pretty neat!

COMMAND:

Now, if you are confident that you want to COMPLETELY remove an application along with it's data, use the command -> **sudo apt purge <packagename>**

Usually, to update and upgrade packages, so that they are new

COMMAND: -> **sudo apt update && sudo apt upgrade**

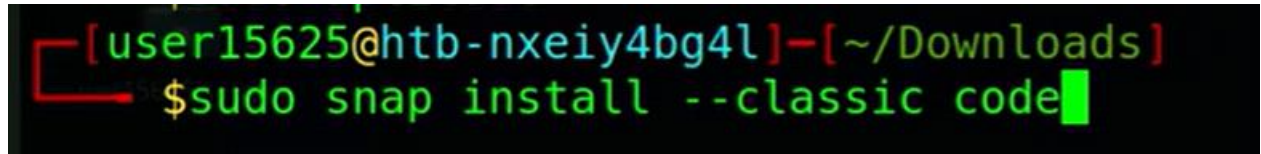
APTITUDE: it is a package manager, it's like apt on steroids. High level
Get into it by -> **Sudo aptitude**
It gives you a GUI in your shell.

SNAP:

It is a package manager, but it doesn't really use the repos, it is like an app store. This is used by new app developers to enlist their apps and make them available to everyone, before they actually get into repos.

Get it by -> **apt install snapd**

Now, using this, install vscode

A terminal window with a black background. The prompt is [user15625@htb-nxeiy4bg4l]-[~/Downloads]. The command being entered is \$sudo snap install --classic code. The cursor is at the end of the command.

```
[user15625@htb-nxeiy4bg4l]-[~/Downloads]  
$sudo snap install --classic code
```

The **--classic** switch is required for it, idk why, look it up.

To add path environment temporarily.

Use -> **export PATH=\$PATH:/snap/bin**

NOW, we get to Python script (python packages / PIP)

PIP is a python package manager

GIT: Online repos storage to access tools or scripts made by other hackers and developers.

Get any repo into your system

Use the command -> **git clone <url of repo>**

Now, to run a python file, you need some dependencies, or requirements, in any python project, a requirements.txt file will be given, you can install the required dependencies by using ----> **PIP**

COMMAND:

pip3 install -r requirements.txt

This tells pip to go inside the txt file and download all the dependencies listed there onto the system.

To run any python file

Use -> **python3** <name of script ending in .py>

That's it about package management and package installation.

For queries, feel free to hit me up on [LinkedIn](#), or via Email at aosawn@gmail.com.

Daemons (Services)

Friday, 19 July 2024 3:32 PM

Your linux system, EVERY linux system, has DEMONS in them.

Well, DAEMONS. Or otherwise known as Linux services.

Daemons are services, that allow your linux OS to actually run.

This lecture will be all about service hunting, enabling disabling starting stopping restarting, ALL of that.

Buckle up, this will be a rough ride.

What are daemons? Officially?

They are the reason everything works the way it works, but they are hidden, working in the background.

Before we get into daemons, we need to know what are processes?

By definition, a process is an instance of a **running program** or a program currently in execution.

Whenever you launch any app, or anything, you start a **process**.
You can check running processes by:

COMMAND:

By using an old command paired with a switch, the **ps** or process status command -> **ps -aux**, this will list all the running processes, now to filter from it, pair it with "**| grep <processname>**" to see only the required process.

Now, the processes that start due to US doing something like double clicking or running, are called **Interactive processes**.

But, when we ran **ps -aux**, it had SO many processes, that we didn't even open, so WHAT ARE THEY?

They are ----- DAEMONS. They are the processes that we didn't start, aka **BACKGROUND Services** or DAEMONS.

Daemons are essential, nothing would work without them,, we have daemons

for printing, networking, heck even SSH.

How do we know if a process is a DAEMON?

A daemon process will have a '**d**' in it's end. Like in the picture below, there is **sshd**

```
Ss  19:22  0:00 /usr/bin/ssh-agent x-se
Ss  19:22  0:00 sshd: /usr/sbin/sshd -D
S+  19:53  0:00 grep --color=auto ssh
```

Now, we move on to the management of DAEMONS.

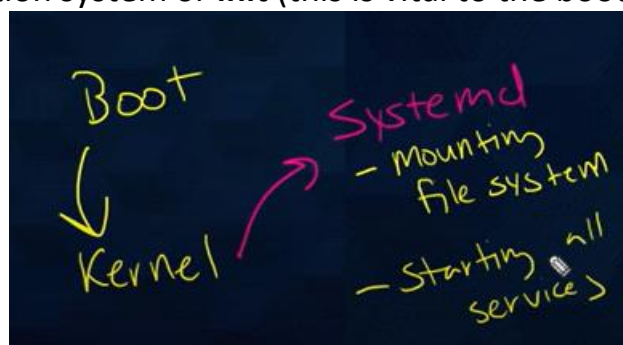
The way we control and manage daemons is by dealing with the MASTER DAEMON. Yes, there is a master to all these little peasants.

His name is "**systemd**". If you wanna do anything with daemons, you have to go through systemd.

SYSTEMD:

Systemd has two jobs, it is a:

1. Service Manager
2. Initialization system or **init** (this is vital to the boot process)



Systemd is the first process that runs when a linux system is booted, then by **forking**, it turns on other services.

COOL COMMAND: A cool way to visualize the process sequence is to use command **ps tree**

This shows how the tree starts from systemd, and all other processes are forked or branched from this.

Systemd process ID will always be ONE (1).

Now, how can we use systemd to control other daemons.

In controlling other daemons using systemd, we use the first job of systemd, that was being a service manager.

*We use a tool called **systemctl***

NOTE: Whenever systemd is talking to its daemons, it refers to them as **UNITS**, so units = daemons, remember that.

Now, let's explore the basic functions of controlling and managing daemons.

STOP: To stop a service, use -> **sudo systemctl stop <servicename>**

STATUS: To check if a service is running or stopped, use -> **sudo systemctl status <servicename>**

START: **sudo systemctl start <servicename>**

RESTART: **sudo systemctl restart <servicename>**

RELOAD: **sudo systemctl reload <servicename>** (not every daemon will be able to run this. Mostly used to just reload config but not the actual service, config like if you've changed it and want the changes to take place without restarting the service.

Now, if you want to see if it can be reloaded, or not you can use -> **sudo systemctl reload-or-restart <servicename>**

This will do either one which is possible.

What if, we want a service to start on boot, or NOT start on boot, what do we do?

We use enable or disable.

sudo systemctl enable <servicename>

sudo systemctl disable <servicename>

```
$sudo systemctl status ntp
p.service - Network Time Service
Loaded: loaded (/lib/systemd/system/ntp.service; disabled; vendor pr
Active: active (running) since Wed 2021-07-21 19:06:49 UTC; 1h 37min
Docs: man:ntpd(8)
Main PID: 827 (ntpd)
```

This means that when the NEXT time the system is booted, this service won't come on. Vice versa for enable.

Now, quick command to see status:

sudo systemctl is-active <servicename>

Another for it -> **sudo systemctl is-enable <servicename>**

Now, HOW TO HUNT FOR DAEMONS?

COMMAND: To see a list of active daemons that the systemd knows about, we type -> **sudo systemctl list-units**

This only shows the ACTIVE ones, still not ALL the daemons.

There are many types of daemons, few to name are service, sockets, devices, mounts etc etc

Now to search or LIST only the daemons of type service, use:

sudo systemctl list-units -t service

Sometimes, daemons break, they don't work, what do you do?

(NGINX is just an example service that we are finding the daemon of, feel free to experiment)

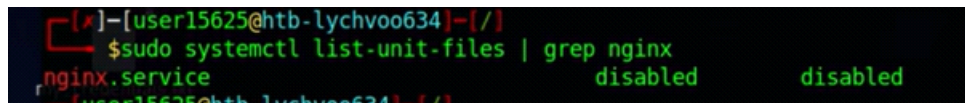
Find it first (harder than you think).

```
File Edit View Search Terminal Help
[user15625@htb-lychvoo634]~[/]
$ sudo systemctl list-units | grep nginx
[user15625@htb-lychvoo634]~[/]
$ sudo systemctl list-units --all | grep nginx
[user15625@htb-lychvoo634]~[/]
$
```

The problem? The above commands ONLY list the units or daemons that have

been parsed into system memory, or loaded.

To fix that, we use the command -> **sudo systemctl list-unit-files | grep nginx**



```
[*]-[user15625@htb-lychvoo634]-[/]  
$sudo systemctl list-unit-files | grep nginx  
nginx.service disabled disabled  
[user15625@htb-lychvoo634]-[/]
```

You can also check status by

Sudo systemctl status <servicename>

That's all.

For queries, feel free to hit me up on [LinkedIn](#), or via Email at aosawn@gmail.com.

Managing Processes

Friday, 19 July 2024 3:34 PM

How do you deal and manage with processes?

In this lecture, we will learn about the management of processes, starting, stopping and all that stuff.

We discussed processes in the previous lecture, now we learn about management.

So the main command for processes is **ps**.

For the ps command, you have to be very specific and tell it what you want to see, otherwise it won't be of any help.

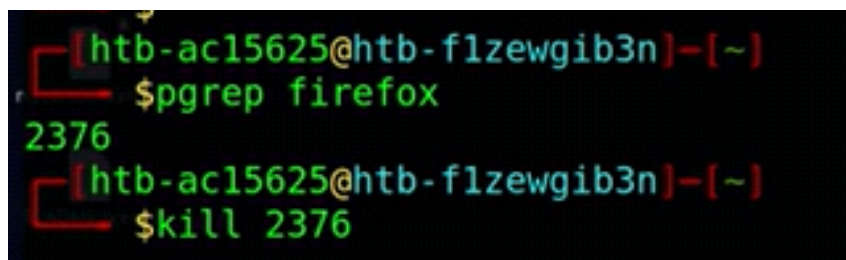
How do you KILL a process?

COMMAND: We use the **kill** command to end any process , but it requires a PID or a JOB ID

There's an easier way to do it, to find it by name

COMMAND: the **pgrep** command will return the PID for the process you name.

Then you can kill it.

A terminal window with a black background and green text. The prompt is [htb-ac15625@htb-flzewgib3n]~. The user enters \$pgrep firefox, and the output is 2376. The user then enters \$kill 2376.

```
[htb-ac15625@htb-flzewgib3n]~  
$pgrep firefox  
2376  
[htb-ac15625@htb-flzewgib3n]~  
$kill 2376
```

PS:

Some commands

Ps --help

Ps --help simple (gives out basic options)

Ps -aux (a-> all users, u=list, x-> all the processes NOT started by this terminal)

COMMAND:

the **top** command , shows all current running processes, and sorts them by their cpu usage (it is a live output)

Even better one is, **HTOP** , it's the same as TOP, but just, prettier.

There are two types of processes:

1. Foreground
2. Background

An example of a foreground process is the **ping** and the **sleep** process or command

(CTRL+C is also a type of KILL command, more on that later)

COMMAND: CTRL+Z when a process is running, this puts a job to sleep

To see how many jobs are sleeping or just the status of them, use command -> **jobs**

We can take a foreground process, and put it into the background

COMMAND: the command **bg** is used to run a job in the background.

NOTE: visually it will still be a foreground process, but you just won't be able to interact with it through the shell now.

We can take a bg process, and put it into the foreground again.

TYPE -> fg 1

This makes it into fg process again.

To avoid doing multiple commands, and just start a process and send it straight to bg, use -> ping -c 300 <website> &
The ampersand sends it straight to bg.

KILL COMMAND: Whenever a kill command is executed, it sends what is called a KILL SIGNAL

To see what kind of signals you can send, list them by using -> **kill -l**

There are a lot of them, but we only care about a few;
Them being:

SIGTERM: Default kill signal, when signal not specified, it is like a , please DIE process, it asks a process to die, and the process can refuse (remember it)

CTRL Z -> uses signal 19 SIGSTOP when pressed once, when pressed twice it uses -> 18 SIGCONT

CTRL C -> uses SIGINT(ERRUPT)

Now, the proper kill command, no suggestion, no request, just killing:

SIGKILL -> This just kills the process forcefully

To kill multiple processes in one go, you can use -> **pkill** with the same switches and instead of PID , use the Pname.

That's all for processes.

For queries, feel free to hit me up on [LinkedIn](#), or via Email at aosawn@gmail.com.

Web Services

Saturday, 20 July 2024 10:38 PM

First up, we are going to be setting up a one command website.

Why?

It's cool && it's useful for hacking

Now, we spin up the website with just one command, using PYTHON. Yeah, that's right, python.

COMMAND:

Python -m http.server, this command will start running a web server, hosted right on your linux machine.

Verify by going to localhost:<portnumber> , the port number will be displayed on the shell.

Go ahead and do it, then come back.

Now, if the port 8000 is not available, you can specify the port number after **Python -m http.server <portnumber>**

The reason the webpage is just showing a list of directories is that it doesn't have an index.html page to look at or consider right now.

Now, we make a new directory on our system from the shell.

COMMAND:

Using the command **mkdir <directory name>**, you can make a new directory wherever you currently are.

Go ahead and make a directory called website on the home of your linux system.

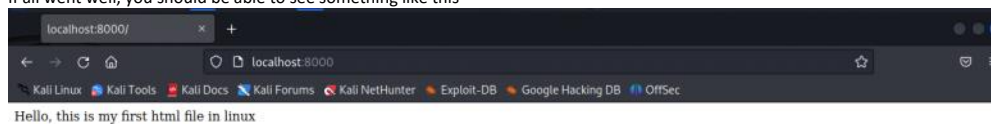
Now, make a new text file there, named index.html, write something in it, basic, and then save it.

COMMAND: use command **nano <filename>** to open a text editor, then in it, write whatever you want, after that, hit Ctrl X to exit, Y to save, and enter to save the file.

Run the webserver again, this time in the directory you made, and go to it.

NOTE: When a webserver is launched by that one liner python code, it will launch in whichever directory you CURRENTLY are.

If all went well, you should be able to see something like this



Now, enough with python, you can do the same thing with your trusty dusty, PHP.

COMMAND: `php -S 127.0.0.1:<portnumber>`

Okay so, what is the IP address in the above command. Well, it's a special IP address that is known as a LOOPBACK address, meaning that it will always refer to the NIC (network interface card) of the machine itself, simply put, it is another way to say LOCALHOST.

But wait, there is moreeee!

We can launch a website with..... **NPM!**
Yes, the node package manager.

COMMAND:

`npx http-server -p <portnumber>`

NOW, it's still not enough, so we see one more way to launch a website.

Using **Apache!**

COMMAND:

`sudo systemctl start apache2`

What does this do?

Well, simply put, it returns the **Response Header** used by the website to establish two way communication.

Kinda like when you send a letter, the mailing details on the outside of the envelope are acting like the response header. Having details on who is sending, from where, and how.



```
10 $ curl -I localhost:8080
10 HTTP/1.1 200 OK
10 Date: Fri, 28 Jan 2022 22:07:51 GMT
10 Server: Apache/2.4.51 (Debian)
10 Last-Modified: Tue, 25 May 2021 22:31:07 GMT
10 ETag: "29cd-5c32f1503c0c0"
10 Accept-Ranges: bytes
10 Content-Length: 10701
10 Vary: Accept-Encoding
10 Content-Type: text/html
10
```

Now, this is the response header, i.e. it goes from the website to the client, where is whatever we sent?

Whatever we send to a website, it goes as a **Request Header**.

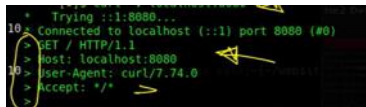
Let's see it.

Run the command:
curl -v <webaddress>

-v means **VERBOSE**, i.e. it will be very vocal, and the output will be lengthy.

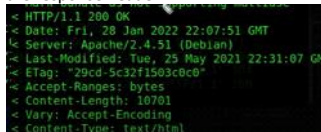
On the very top of the output, you will see some stuff, that is the request header, and the bottom of it will be the response header. But, how do we differentiate?

All the text that starts with an arrow pointing right, that is part of the request header.



```
10 * Trying ::1:8080...
10 * Connected to localhost (::1) port 8080 (#0)
10 > GET / HTTP/1.1
10 > Host: localhost:8080
10 > User-Agent: curl/7.74.0
10 > Accept: */*
10
```

Similarly, all the text that is followed by an arrow pointing left, that is part of the response header.



```
10 < HTTP/1.1 200 OK
10 < Date: Fri, 28 Jan 2022 22:07:51 GMT
10 < Server: Apache/2.4.51 (Debian)
10 < Last-Modified: Tue, 25 May 2021 22:31:07 GMT
10 < ETag: "29cd-5c32f1503c0c0"
10 < Accept-Ranges: bytes
10 < Content-Length: 10701
10 < Vary: Accept-Encoding
10 < Content-Type: text/html
10
```

There are various types of request methods, whenever a client is communicating with a website, the first one will be a GET request. That means you are telling the website to GET you some webpage.

If it works, all good, then the website sends you a response having 200 OK in the response and then telling you what it is sending.

That's that.

For queries, feel free to hit me up on [LinkedIn](#), or via Email at aosawn@gmail.com.

Hacks (To be fast in Linux)

Sunday, 21 July 2024 12:05 AM

In this bonus lecture, we will be looking at some hacks to just overall increase your efficiency when using the shell.

The lesser you use your mouse during a job, the more time you will be able to save.

1. Cd -> changing directories, the hack being when you just write cd with itself, you quickly jump back to the home directory, no matter where you were.
2. Pwd -> Tells you where you currently are, what is your current working directory.
3. If you want to just go back ONE directory, use -> **cd ..**
4. If you want to go all the way back to the main directory, use -> **cd ../../**, this just means to take 2 backs, pairing them.
5. Clear -> cleans your terminal, or CTRL + L does the same.
6. You can use your directional arrows on the keyboard, up and down arrow, you get PREVIOUS commands that you have entered, kinda like a history for commands.
7. Suppose you were super deep in a directory, and you just came on home, but you forgot something and want to go back into the previous directory quickly, how do you do that? Use -> "**cd -**" to quickly go back to the previous directory.
8. "cd -" refers to the variable "**\$OLDPWD**", which you can also see using -> **echo \$OLDPWD**
9. "**ls -l**" displays all the items in a formatted list, along with some other important stuff like the read write execute permissions for each file and folder and some metadata.
10. "**ls -a**" this shows HIDDEN files along with the normal files, all the hidden files will be starting with a DOT (.).
11. "ll" and "la" also do the same stuff as above, saving you more time. These are aliases, like we created shortcuts for our commands.
12. CTRL + A sends you to the start of a command, CTRL + E sends you to the end of it, handy when you're writing a long command and don't want to use arrow keys to go back and forth to fix a typo.
13. CTRL + U deletes everything before your cursor, you don't have to backspace it all.
14. CTRL + K is the opposite of the above, it deletes everything after the cursor.
15. If you mistakenly delete a command, hit CTRL+Y to get it back.
16. Hit CTRL+X+E to open whatever is written in your shell, in a text editor,

this will allow you to quickly edit really long strings of commands.

17. To look at a file, we used cat command to view them, but it doesn't really do well with extensive files, e.g. log files. So, what do we do? We use "**less**", it doesn't load the entire file, it only loads parts of it as they are seen.
18. If you forgot to write sudo with a command, just write "**sudo !!**" this will automatically sudo the previous command and run it.
19. A command called "**tail**" is useful for when you are analyzing log files and you ONLY want the most recent entries or logs, tail brings up the recent 10 lines of the file.
20. What if you forgot a path while writing a directory path, you can just write the starting directory, e.g. /var/ then press "TAB" key twice, and it will give you all available options to go from var.
21. What if you want to look at logs in real time, right after tail, write "tail -f", this updates logs live.
22. You can search your command history, by entering CTRL + R, and that will open a reverse search in the Shell.

That's all for this one.

For queries, feel free to hit me up on [LinkedIn](#), or via Email at aosawn@gmail.com.

Directories and Files

Sunday, 21 July 2024 12:43 AM

To create a new file

COMMAND:

touch is used to make a new file of any type, in any directory

To create multiple files

Use touch with multiple file names

e.g. touch 1 2 3 4

This will make 4 files named 1 2 3 4 respectively.

These files are empty by default.

We used **cat** to view file contents, but we can also make a file with stuff inside of it using cat.

e.g. -> cat > file.txt , then the shell will wait for your input to put it into the file

Helpful when writing bash scripts.

To create an EOF triggered file, i.e. a file that will end when a certain symbol or word is entered into the terminal, we use the command

cat << EOF > nameoffourfile.txt

This EOF can be anything, a symbol, a word, a number, this means that the file will keep taking input, until this EOF is entered

To put stuff into a file in just one line

Do

echo "whatever you want to write" > nameoffile.txt

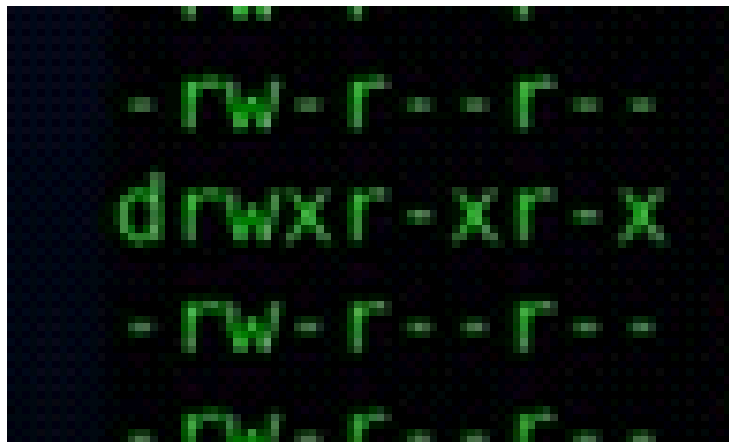
This will quickly send whatever is echoed into your file

Now, moving onto directories:

To make a directory -> **mkdir <nameofdir>**

When you do "ls -l" and you see all the files and dirs listed, how do you differentiate with what is a file and what is a directory?

- a. It can be color coded, like directories in one color and files in a different
- b. It is differentiated by the permissions column, a directory starts with a "d" and a file starts with a "-"



You can make multiple directories in one single command, just like touch, try it yourself.

Now, moving onto moving files,

COMMAND: mv moves files, syntax being

mv <filetobemoved> <path of dir where file is to be placed>

Also, you can move a file whilst simultaneously changing it's name when it will be placed in the new location, just after the path of the dir, write a new name for it like

Path/newname.txt

Try and experiment with removing files and directories, using simple **rm** and **rmdir** commands, along with **rm switches**.

That's all.

For queries, feel free to hit me up on [LinkedIn](#), or via Email at aosawn@gmail.com.

