Linux

1.Facts:

- Use HackTheBox Academy to use Linux on Browser
- Linux is basically Linus Torvalds + Unix
- Linux is technically NOT an OS but a Kernel (usually we call it OS for simplicity), and OSes are built on top of that Linux Kernel are called distributions or distros or flavors of Linux. Eg: ParrotOS, Kali Linux, Ubuntu, etc.
- It's more secure (very less viruses), faster and most importantly open source (so we can make our own OS i.e., distro) & free
- In Windows Subsystem for Linux (WSL) goto cd /mnt/c (To go to C:), Make sure below are satisfied before using WSL:
    - CPU Virtualization is enabled (Performance TAB in Task Manager, if not enable it in BIOS)
    - Hyper-V is on (Win feature)
    - Virtual Machine Platform is on (Win feature)
    - Windows Subsystem for Linux is on (Win feature)
    - Go to Microsoft store and download Kali Linux or Ubuntu
- If a command gives access denied error, try adding sudo (super user do) prefix.
- Files having dot(.) at beginning means that file is a hidden file.
- Etsy /etc/ contains configuration files, group info, sudoers file, hashed passwords, etc. in Linux.
- Maximum size (in bytes) of the filename in Linux is 255 bytes
- Script command is commonly used to record sessions in Linux.
- ntfs file system that is used to access remote systems.
- Samba is basically an open-source software suite. It runs a number of different Operating systems such as OpenVMS, IBM, etc. It is generally used to connect Linux machines to Microsoft network resources simply by providing Microsoft SMB support. It provides more secure, stable, fast file and print services for every client or user using SMB (Server Message Block) or CIFS (Central Server Message Block) protocol.
- 3 standard streams in Linux: stdin, stdout, stderr
- ext2 file system type is used for Linux Systems.
- There are two commands to schedule tasks in Linux: cron and at.

- [Y/n] option capital means the default value.
- Linux only work on Line Feed (\n) while Windows uses Carriage Return Line Feed (\r\n) this can cause issues so make sure to add this in git.
- mkcd(){

  ```
  mkcd(){
      mkdir -p "$1" && cd "$1"
  }
  ```
  in .bashrc will make and cd to new dir

2.Terminal Stuff:

- It's not actually terminal but a terminal emulator
- An actual terminal was a device with monitor keyboard attached to it and used back in the day to interact with computer eg: VT100
- The black window GUI what we have is terminal emulator it pretends to be a terminal
- It has one job i.e., to run commands so that we can interact with PC
- We can call it a shell, a console or even a terminal for simplicity
  - There are various types of Shell most popular is BASH (Bourne Again Shell), others exist too like PowerShell, C Shell, Korn Shell, Bourne Shell, etc., you can check name of shell using ps (proc status) command
- BASH is case sensitive while DOS of Windows is not.
- Use Ctrl+Alt+T on Ubuntu for Terminal
- *echo "Hello I'm Terminal"* (Prints this on terminal)
- Can use Pipe Command here too to do stuff together

  ifconfig | grep "inet"
- ./demo.xyz will run(execute) demo.xyz file present in same directory (./)
  - bash demo.sh (will run the bash file) sometimes ./bash.sh will give permission denied error, most likely we need chmod
  - python3 bomber.py (will run the python file)
- We always see **user@hostname:current_directorySYMBOL** in our Terminal
- Where SYMBOL can be $ for User or # for root user, has godlike powers
  - use *sudo -i* or *sudo su* (switch user) to become root user
  - CTRL+D to logout as root or use *su kartik* or type *logout* to switch user from root to kartik, CTRL+D can also be used to exit out jobs/terminal.

- NOTE: The root account is disabled by default in Ubuntu, so there is no root password, that's why *sudo su* fails with an authentication error.
- Use *clear* (clear's up terminal or CTRL+L) to clear terminal, NOTE: CTRL+L doesn’t actually clear the terminal but actually hits Enter and pointer is at top of screen.
- *which $SHELL* (tells which shell we’re using)
- Alt + 1, 2, … to change tabs on the terminal (also works on browsers)
- Ctrl + C to terminate a running command. (Interrupt)
- Ctrl + A/E to move pointer to start/end (Can use Home and End on Keyboard too)
- Ctrl + Zoom (or Pinch) (increase/decrease size of terminal)
- Ctrl + Shift + Zoom (or Pinch) (increase/decrease transparency)
- Ctrl + U/K (obliterates everything before/after pointer)
- Ctrl + Y (paste obliterated content where the cursor is)
- Ctrl + X + E to transfer the current command to nano editor
- Ctrl + D to exit or logout (can move from one user back to original user too)
- Since Ctrl + C is used for interrupt, we use Ctrl + Shift + C for copy and C+S+V for paste.
- To open Tab use Browser’s logic + Shift, New Tab-> Ctrl+Shift+T, so on…
- <command1> && <command2> && … (executes all L to R bitwise and)
- <command1> ; <command2> ; … (executes all L to R bitwise or)
- Alt + Backspace (remove words, in MS Word use Ctrl + Backspace)
- Single Tab (to get autofill)
- Double Tab (to get list of all autofill)
- Ctrl + R (reverse searches the commands we’ve done so far, keep doing Ctrl + R if not satisfied with output)
- When seeing multiple pages in git log or systemctl list-units use down arrow or Pg-Down to move and press q to exit it.
- *sudo apt purge googler -y* (Automatic yes for [y/n] choice)

3. Directory & File Stuff:

- *pwd* (Print working directory)
- *cat ./file.txt* (will cat file.txt present in same directory)

    - Although we can use *cat file.txt* too as it looks in same directory as well
    - *cat /file.txt* will give error because there is probably no file named file.txt in root (/) so use cat ./file.txt
- .. stands for parent directory and . stands for current directory
- *ls* (list fies, directories in current dir)
    - *ls ./newfolder* (lists contents of newfolder that's in curr dir)
    - *ls subfolder* (lists contents of subfolder)
    - -t (sorts file based on time created)
    - -l (long info)
    - -a (all info even hidden files)
    - -lt (long info time sorted)
    - -r (reverse order)
    - *ll* (alias for ls -la)
- *nano demo.txt* (creates text file and opens nano editor)
    - CTRL+O Enter to save & CTRL+X to exit
- *vim demo.txt* (creates text file and opens vim editor)
    - Press 'i' to insert in file i.e edit.
    - After you're done modifying press ESC
    - Enter ':wq!' to save and exit (! Is optional)
    - Enter ':q!' to exit without saving
- *mkdir Folder1* (makes directory)
    - -p (for making us parent directory and we can create child directories), *mkdir -p directory/anotherone/anotherone/another*
- *tree* (To see the directories in a tree format)
- *rmdir Folder2* (deletes folder2 but must be empty for security reasons)
- *rm -r Folder2* (deletes folder2 and its contents, -r means recursive)
- *rm jojo.txt* (deletes file)
    - *rm dave.txt pic.jpg* (deletes multiple listed files)
- *cp <fileSRC> <desti>* (copies file/directory, way to rename too)
    - Let's say we want to copy ./dir/anotherone/coolstuff to ./dir/anotherone/anotherone/coolstuff . We'll have to do it recursively using *cp -r ./dir/anotherone/coolstuff ./dir/anotherone/anotherone/coolstuff* because it has child directories.
- *mv <fileSRC> <desti>* (moves file/directory, a way to rename too)

    - *mv filea.jpg fileb.txt gamer.conf ./secretfolder* (moves 3 files to secretfolder which is a subfolder in our current directory that's why we used dot before it)
- *sort <file>* (sorts the content of file by ASCII)
    - -u (unique), -r (reverse sort, DESC), -n (numeric sort), -c (check if sorted)
- *locate sample.txt* (locates sample.txt file)
- *touch gta6.jpg* (creates dummy file)
    - *touch gta{1..6}* (you guessed it)
    - *touch -d tomorrow gta7.txt* (File date will be that of tomorrow)
- cat demo.txt (shows what's in txt file, short for concatenate)
    - .z extension means file is compressed so use *zcat secret.z*
    - *cat filea.txt fileb.txt* (concatenate files and prints on terminal)
    - *cat > filename.txt* (creates a new file and can edit on terminal itself, use CTRL+C to save and exit)
    - *cat filename1.txt filename2.txt > filename3.txt* (appends filename2.txt to filename2.txt and stores the output in filename3.txt)
    - *tac demo.txt* (shows contents in reverse order, i.e last line on top)
    - *cat << EOF > ourfile.txt* (Will take input to write in file, until we type EOF & hit Enter, this EOF can be anything like END, etc.)
- *echo "Gay ass nigga" > gay.txt* (same in windows)
    - *echo "2nd line" >> gay.txt* (appends in gay .txt)
- *cd* (change directory, goes to /home/username i.e ~)
    - *cd ~[username]* (goes to another user's home dir)
    - *cd ~ (goes to current user's home dir)*
    - *cd /* (goes to root directory)
    - *cd ..* (moves one directory up)
    - *cd ../..* (goes back 2 directories)
    - *cd -* (moves to your previous directory, see echo $OLDPWD)
    - *cd david* (changes directory to david)
- Useful echo commands:
    - *echo $PWD* (shows current directory in cmdline)
    - *echo $OLDPWD* (shows old directory in cmdline)
    - *echo $RANDOM* (generates a random number b/w 0-(2^15-1)
    - *echo $SHELL* (tells which shell we're using eg:/bin/bash)
        - can also use *ps* or see in /etc/passwd file of user

    - *echo $USER* (tells the username)
    - *echo $HOSTNAME* (tells the hostname)
- Files have 3 mode read4, write2, xecute1 useful in chmod
    - *chmod u=rwx, g=rx, o=r filename.txt* (change mode of user=rwx group=rx other=r)
    - *chmod 754 filename* (user=rwx group=rx other=r)

- **Everything in Linux is a goddamn FILE!**
    - Folders, hard drives, ram, router info, even commands
    - You can check command as files in (/bin and /sbin (sudo commands))
    - You can check HDD/RAM as files in (/dev short for devices)
    - You can check config/network files in (/etc called etsy)
    - We can copy the file and rename the new ones as our need. So this is also a stupid way of alias by renaming a command
    - There's bin and sbin in /usr too infact /usr/bin or /usr/sbin has more commands than /bin or /sbin but still has huge overlapping. Why is it so? It has old history that has something to do with hard-disk space.
    - If we want to see which folder is being used for command 'clear' we can type *which clear* most likely we'll get '/usr/bin/clear'
- **DELETE EVERYTHING COMMAND:** *sudo rm -rf –no-preserve-root /* (means super user remove recursively forced and don't preserve root)
- Directories in / (root):

    bin (stores the binaries i.e commands)
    sbin (sudo binaries)
    home (home directory it has all users of the system)
    boot (dir that has boot config files)
    var (web app files, log files)
    dev (devices details)
    tmp (temproary files)
    usr (lot of similar files of root but works locally-ish)
    etc (et cetra folder it has all configurations of n/w, user details, hashed passwords, sudoers file, etc. called etsy)
    lib (library folder that has imp files for sys)
    mnt (mounted devices by user explicitly)
    media (mounted devices by OS implicitly, like pen drive, etc)

- *which ls* (will tell us which binary’s ls we’re taking about i.e., /bin/ls or /usr/bin/ls. Usually it’s **/usr/bin/<command>**

4. User Stuff:

- *whoami* (prints who are you, i.e., username you’re logged in as)
- *id* (identifications)
- *groups* (tells you what group(s) we are the member of)
- *w* (Detailed version of all users logged in to system)
- *who* (Truncated version of "w", tell who else is logged into the system)
- ***cat /etc/passwd*** (lists all users), lot of them are non-login users who server different purpose like root, daemon, games, sync, etc.

  x means the user's password is stored in other file (shadow file, **/etc/shadow**), if you investigate it, it'll give a hash-function of password

  - 1 is MD5.
  - $2a$ is Blowfish.
  - $2y$ is Blowfish.
  - 5 is SHA-256.
  - 6 is SHA-512.
- *adduser Thor* (more formal and powerful, needs sudo)

  When we add a user we also add a group, so it'll be like thor:x:1001(user-id):1001(group-id):blah-blah:/home/thor(home-dir):/home/bash(default-shell)

- *useradd John* (doesn't give home directory (although /etc/passwd says it has but it doesn’t we can check by ls the /home directory), no password, (again the passwd file says x in front but if we cat the shadow file it has no hash value) default shell is sh NOT bash, i.e lazy)

- *sudo passwd ironman* (to set password for ironman user, note: you cannot su ironman without ironman having password coz it will ask for password and anything you do will result in Authentication Failure)
- *usermod ironman --shell /bin/bash* (sets shell to bash)
- *usermod -l tonystark ironman* (renames ironman to tonystark)
- *usermod ironman -m* (gives home directory)
- *userdel John* (Deletes user John)

- *su [options] [username [argument]]* (switch user)
    - *sudo su* (to switch to root user #)
    - *su thor* (switch to user thor, use CTRL+D to logout of thor)
    - -shell or -s (diff shell environment to run)
    - -login or -l (runs a login script, needs passwrd ofc)
- If we switch user to thor by -su thor then do -sudo useradd mohan, it'll ask for thor's password then we'll get error "thor is not in the sudoers file. This incident will be reported.". So, this means kartik was in sudoers file, but others were not.
- Sudoers file (/etc/sudoers) has list of users/group that have sudo ability, it's very likely that we can mess up the file so there are lot of checks to make sure we don't ruin it.
    - We can use nano/vim to edit it BUT best practice is to use *sudo visudo* instead to modify the sudoers file so that there are lot of checks in place.
    - Under **#User privilege specification** there are name of users who can use sudo, ofc there will be root ALL=(ALL:ALL) ALL, we can add any other user too.
    - Say we want thanos to have unrestricted access to add user we can do: **thanos ALL=/sbin/useradd** but we don't want that so we can do: **thanos ALL=ALL** for giving him access to all commands.
- *sudo groupadd infgauntlet* (adds group called infgauntlet, checkexistence in **/etc/group** NOT in /etc/passwd)
    - We can add this user to sudoer's file using sudo visudo
    - We can see the section #Allow members of group sudo to execute any command we can already see %sudo there we can put %infgauntlet ALL = NOPASSWD:ALL (meaning we can execute all commands without password)
- Use *sudo usermod -aG infgautlet ironman* (will appendGroup infgautlet with ironman) we can verify by catting /etc/group, thor will have its name there.
    - Also add thanos to infgautlet
    - In the end ironman wins and delete thanos from infgautlet group, using -sudo gpasswd -d thanos infgautlet
    - Now ironman can add anyone he wants and can delete thanos too.

- Ironman realizes infgautlet group is too powerful so he deletes that group using sudo groupdel infgautlet. (doesn’t deletes the users in the group just the group only)
- Now even if ironman wants to, he can’t do any sudo stuff anymore.

5.Internet Stuff:

- *ping www.google.com* (To check if server is up and running, i.e., talk to me)
- *nslookup myip.opendns.com resolver1.opendns.com* (yes it works here too!)
- *nslookup www.youtube.com* (DNS lookup for website)
- *traceroute www.google.com* (same as tracert in windows, needs to be installed on Linux)
- *netstat -aon* (yes it works here too!)
- *ifconfig* (Interface configuration, do sudo apt install net-tools)
  - -a (all info, including MACs and Maximum Transfer Unit (MTU))
- *iwconfig* (Wireless configuration, sudo apt install wireless-tools)
- *hostname [option]* (to know the system’s hostname)
  - -a or –alias displays the hostname’s alias.
  - -A or –all-fqdns displays the machine’s Fully Qualified Domain Name (FQDN).
  - -i or –ip-address displays the machine’s IP address.
- *curl wttr.in/delhi* (Curls the wttr.in/delhi to terminal that’s actually a weather website)
- *arp* - To view kernel's ARP table (Address Resolution Table)
- *mtr* - Combination of ping and traceroute
- *whois* - Get detailed info of website (sudo apt install whois)
- *tcpdump* - Captures traffic that's passing thru n/w interface (IMPORTANT)
- *ssh mike@192.168.155.65* (logs at given IP as user mike, Secure Shell)
  - shell is an environment that can run prog, commands or scripts,
  - SSH is a protocol used to securely(encrypt) connect to a remote server/system runs at TCP/IP port 22.

- We can deploy a website/sever with help of python using *python -m http.server* We'll get website on 0.0.0.0 port 8000
    - When we'll visit it by localhost:8000 on browser we'll have directory listing and each access will show up on our terminal.
    - We can specify our own port too *python -m http.server 1337*
    - We can only access it on same network to access it globally we can sign up on ngrok service, use *ngrok http 1337* to tunnel our 1337 port to http. We can also use ngrok tcp 3306 to tunnel or host our localhost mysql database online.
    - The reason it's showing directory that's because it doesn't have index.html to show, so in the same folder create index.html file put something in it and BOOM! It'll be shown.
- We can deploy a website/server with help of php using *php -S 127.0.0.1:8085* (which is nothing but localhost:8085, localhost is DNS for 127.0.0.1 which is a loopback address that points back to our NIC card)
- We can deploy a website/server with help of npx using *npx http-server -p 8086*.
- We can deploy a website/server with help of apache and systemctl.
    - Use *sudo systemctl start apache2*, we'll most likely get an error and will be told to check journalctl -xe (You know what to do)
        - *sudo systemctl restart systemd-journald* (How did I get its name? see Daemon section, it'll restart journal service)
        - *sudo journalctl -xe* (To view log files, we'll see port 80 is already in use)
        - We just have to change the default port 80 to something other like 4200 edit the file using *sudo nano /etc/apache2/ports.conf*.
        - Edit 'Listen 80' to 'Listen 8080'.
    - Now *sudo systemctl start apache2* will work! And when we'll open localhost:8080, we will have default page of apache.
- Talking to website using command line (wget, curl and etc.)
    - curl stands for client URL, it's a powerful tool that's implemented on lot of devices and can not only download files but also communicate with websites/servers and so on…
        - *curl localhost:1337* (It'll display website's index HTML code in console)

        - *curl -o coolwebsite localhost:8080* (downloads the website, then use cat coolwebsite to see its contents)
        - *curl -I localhost:8080* ({dash capital i} will tell us status code, date, server type, etc. i.e., a Response Header)
        - Let's look at the Request Header now *curl -v localhost:8080* (-v for verbose), we'll get Request Header (> denoted by right arrow, usually GET request), Response Header (<) and HTML code.
    - *wget localhost:1337,* it'll download the file index.html in the working directory and some headers working like HTTP request send, awaiting response 200 OK.
- **Status Code Definitions:**
    - 100 Continue
    - 200 OK
    - 302 FOUND
    - 400 Bad Request
    - 403 Forbidden
    - 404 Not Found
    - 500 Internal Server Error
- *nmcli connection show* (Shows known wi-fi)
  cd /etc/NetworkManager/system-connections/
  cat Finnovation.nmconnection (shows password)

6. Processes and Services Stuff:

- *ps* (proc status, snapshot)
    - TTY is a computer terminal. In cmmd ps, it's a terminal that xecuted a particular cmmd, stands for "TeleTYpewriter", which were devices that allowed users to connect to computer way back.
    - Running it alone will only give 2 results the shell name and ps command itself.
    - *ps -u kartik* (lists all the processes of user kartik), it'll have ton of processes, so to list our required process like firefox use *ps -u kartik | grep firefox*.
    - *ps -aux* is most useful it tells allusers, username and x for all the other processes that are xecuted without this terminal.

- *kill 3848* (kills the process with PID 3848, the kill command can only kill if we have PID NOT by name)
- *pgrep firefox* (process grep command will return the PID for firefox
- *top* (shows running processes by CPU usage in real time)
- *htop* (shows much more detail than top, i.e., top on steroids)
- There're 2 types of processes the foreground and background processes.
- Like *ping -c 100 www.google.com* (pings google count=100 times) it's actually a foreground process and happening in front of us, so either we wait for it to finish or we interrupt it by CTRL+C (which is actually a version of kill)
- Like *sleep 30* (sleeps for 30 secs) we can't do anything else other than interrupting it or waiting 30secs.
- We can put *ping & sleep* and put them in background by- CTRL+Z to knock the job to sleep (cross check using *ps -ax* and see T in front of stopped proc) and can see which jobs are running/stopped using *jobs* command in console, we'll see something like [1]+ Stopped   <cmd>. Now to put it to background type *bg 1*. It'll run the job with ID 1 as bg process, if we only run *bg* it'll run the only process in jobs list. After it's running, we won't be able to stop it (Do CTRLZ, CTRLC nothing will work). So, during that chaotic running type *fg 1* to convert it into foreground process and then use CTRLC to end it or you can simply do *kill %3* if Id=3.
- If we want to put a job directly to background use *ping -c 100 www.google.com &* (i.e put '&' in the end)
- When we use kill command it sends the kill signal, and there are many kill signals use *kill -l* (to list all kill signal list)
  - Default signal is 15 (SIGTERM) – A suggestion to proc i.e plz die and process can refuse too.
  - CTRL+Z is 19 (SIGSTOP) and if we use CTRL+Z again it's 18 (SIGCONT)
  - CTRL+C is 2 (SIGINT) – A signal interrupt
  - Signal 9 (SIGKILL) – Forcefully terminates anything
- Run *sleep 900 &* then it'll run in background use *ps* to find its process ID then do *kill -19 1234* (does CTRL+Z to sleep 900, i.e stopping it) or do *kill -2 1234* (does CTRL+C to sleep 900, i.e Interrupting it)
- Just like ps had pgrep command, kill has pkill command, here we don't need a process ID, we can just type a freaking name.

- *pkill -9 ping* (Force kills all the ping processes)
- *mount -l* (list of mounted devices)
- *service <servicename> status* (checks if particular service is running)
- *service --status-all* (check the status of all the services)
- *service <servicename> start/stop* (start/stop a service)

7. Linux Package Management:

- Anything we want to install is contained in packages, to install them we have package managers, 2 mains are: dpkg (de-package) and apt (advanced package tool).
- File extension for packages in Linux is .deb specifically Debian based. Other distros like RedHat, Centos, openSUSE we'll have .rpm
- Windows has .exe Mac has .dmg
- About dpkg:
    - It is low level pkg manager and is kind of dumb.
    - 1st flaw: in apt we did sudo apt install xyz but here we have to download the .deb file first from website of xyz application.
    - Then do *sudo dpkg -i discord-0.0.15.deb* (-i for install ofc)
    - 2nd flaw: We'll get shit ton of dependency errors, as most pkges need other pkges to be installed to work and dpkg doesn't automatically does that.
    - *dpkg -l* (lists all the installed packages)
- About apt:
    - It is high level pkg manager and is kind of smart.
    - *sudo apt update* (to keep our repository up to date)
    - *sudo apt install sl* (that's it, it'll install steamloco package from link in repository and if some other dependency exists it'll install them too.)
    - NOTE: The repositories are updated regularly so before installing using *sudo apt install xyz* make sure to do *sudo apt update*
    - Use *sudo apt edit-sources* to get what repo we are using. It's usually located at **/etc/apt/sources.list.d/parrot.list**, we can cat or nano it to view.
    - It'll have deb https://deb.parrot.sh/parrot/ this is where it gets the package details. Open the link we'll have index of repos.

- This is the list provided by parrotOS, sometimes we want to install something that's not in parrotOS repo, so what we can do is most of time they'll have their own repository that we can add to our sources list. Do *apt update* and *apt install* and we're done.
- *Sudo apt list* will list all the pkges in our repositories.
- *Sudo apt list –installed* show installed pkges (can use grep too)
- *Sudo apt show nmap* (shows what nmap does, i.e description)
- *Sudo apt remove discord* (to remove the application NOT user data).
- *Sudo apt purge discord* (removes everything)
- Quite a lot of times our apps get updated and we want that so we do *sudo apt update && sudo apt upgrade,* it'll update the repo so that if we have new version of any app, we'll know it and if it is then upgrade the app.
  - *sudo apt update && sudo apt full-upgrade* will uninstall the preexisting version of any application whose update has been found and install the fresh new one unlike normal upgrade.

- About aptitude:
  - It's another pkg installer
  - High level and like apt on steroids.
  - *Sudo aptitude* to use it, it has GUI like interface to make stuff simpler.
- About snap:
  - It's called snapd, it's not like a repository but more like a store
  - Similar working to apt considering command line.
  - So, developers putting their apps on repositories take lot of time, so this comes in handy where devs can just put their app on snap store and BOOM! It's easily available to every Linux distro in a snap.
  - We need to install it first, do it by *sudo apt install* snapd
  - Syntax: *sudo snap install –classic code (code is pkg name for vscode)*
- *Pip for python or gem for Ruby:*
  - It's program oriented pkg installer.
  - Usually, program has requirements.txt file to install it use *pip3 -r requirements.txt*

    - Then use python3 file_name.py
- Git clone:
    - To install stuff from github
    - *git clone* https://github.com/TheSpeedX/TBomb will download the Folder in the current directory.

8. Daemons (Day-men):

- A daemon is a service process that runs in the background and supervises the system & provides functionality to other processes. Basically, it helps the OS run the way it's supposed to be running.
- *ps -aux* (Lists all the running processes with PID, %CPU, %MEM, TIME, etc.
- Say we've sublime running and we want to see detail use *ps -aux.*
- It's like background processes for windows.
- We have daemons for printing, SSH, networking, etc.
- To distinguish between a normal process and a daemon, there's difference in name. A daemon process has 'd' at the end of process name. Cross check by typing *ps -aux | grep ssh* we'll see a daemon named "sshd".
- The reason they're called daemon is because in Greek mythology Demon was a supernatural entity with no biasness with good or bad, so they named these powerful automatic running background processes as daemon.
- There is a **Master Daemon** called system, he is service manager and most importantly does initialization of system (init).
    - Flow: PC On->CPU initializes itself and looks for UEFI in the BIOS chip->BIOS initializes hardware and runs POST->BIOS transfers control to OS's bootloader located at MIBR of HDD/SSD->Bootloader loads up the Kernel->systemd is initialized (init) it performs mounting file sys, start all bg processes, etc.->Fork other daemons.
        - GRUB (GRand Unified Bootloader) for RedHat Linux
        - LILO (LInux LOader) for most of Linux based OS
    - *pstree* will create processes as a tree, notice how systemd is root.

    - Since it’s the very first process/daemon it’s PID is 1. Cross check by *ps -aux*.
    - systemd isn’t the only init process out there they are others too but he’s the main one we see on any modern linux distro.
        - sysv, upstart (old deamon of linux), openrc, etc. are few other init processes. But thankfully most if not all linux distros have adopted systemd as their init process.
- Systemd calls other daemons as units.
- *systemctl* command is use to manipulate daemons (start,stop,restart,etc.)
    - **Stop-** *sudo systemctl stop sshd*
    - **Status-** *sudo systemctl status sshd* (see Active or Inactive)
    - **Start-** *sudo systemctl start sshd*
    - **Restart-** *sudo systemctl restart sshd or sudo systemctl reload sshd or sudo systemctl reload-or-restart sshd*
    - **Disable-** *sudo systemctl disable ntp* (Disables Network Time Protocol to start at startup, cross check by seeing status we’ll have ntp.service; disabled;)
    - **Enable-** *sudo systemctl enable ntp* (Enables ntp to start at startup)
    - **One-liner-** *sudo sytemctl is-active ntp, sudo sytemctl is-enable ntp*
- *systemctl suspend* (sleeps/hibernates PC)
- To see list of active daemons, use *sudo systemctl list-units*
    - Notice how ntp’s actual name is ntp.service but when we stopped, restarted using only start ntp, or stop ntp, systemctl is smart enough to know ntp is actually a service daemon so it understands we meant ntp.service daemon.
    - There are various types of daemons like sockets, mounts, devices, etc.
- To see list of active services daemons, use *sudo systemctl list-units -t service* (-t for type)
- To see list of all daemons (active or inactive), use *sudo systemctl list-units --all* (but this will still list daemons that are parsed to memory, any daemon which is not parsed will not be here)
- To list every daemon (inactive, active, parsed, unparsed) use *sudo systemctl list-unit-files*
    - *Sudo systemctl list-unit-files | grep nginx*

- If we hit a failure in systemctl it will tell us to check “journalctl -xe” which is a log file for systemd stuff.
    - Try *sudo systemctl start ngix* (we’ll get an error and a recommendation to see log file)
    - *sudo journalctl -xe* (There will be no entry here too! that’s because journalctl is itself a daemon which we need to restart)
    - We’ll need full exact name for journalctl, so use *sudo systemctl list-units | grep journal* and we’ll have a list, since it’s a service we must select journald.service.
    - *sudo systemctl restart systemd-journald* (To restart logging).
    - Now we can run *sudo systemctl start ngix* we’ll ofc get error but logs will work now.
    - *sudo journalctl -xe* (We’ll see that ngix uses port 80 and it’s already in use). How do we fix that? See Internet Stuff section.

9. Others Commands:

- {command} > output.txt (writes output of command to file)
- *sudo !!* will run the prev command with sudo (not only limited to sudo)
- *!$* will take the input parameter from previous command
    - *echo "bhag saale"*
    - *cowsay !$*
- *wc test.cpp* (tells #ofLines, #ofwords and size(bytes) of file, stands for word count), also works with commands using pipe
    - -l (lines)
    - -m (chars)
    - -c (byes)
    - -L (max line length)
    - -w (words)
- *uname* (OS Name by default)
    - -a (all)
    - -s (kernel-name)
    - -v (Kernel-version)
    - -r (kernel-release)
    - -p (processor)
    - -o (OS, default)
- *lsb_release -a* (Tells sys version, jammy etc.)

- *free -m* (how much memory is being used)
  - free itself shows total, used, free, shared, cache, etc.
- *exit* (exits the system, can use CTRL+D too)
- *man <command>* (gives manual for command, also info <command>, if want short use --help)
- *lpr demo.txt* (sends file for printing)
- *passwd* (for changing password of current user)
- *grep "linux" interview.txt* (GlobalRegularExpressionPrint, searches word "linux" in file)
  - *ifconfig | grep "inet"* (looks for inet in ifconfig, just like cmd's | findstr)
- *tar -cvf filename.tar.gz* (tar is used to create and xtract archive files,Tape ArchiveR)
  - -c(create), -v(verbose) -f(file)
- *rev* (Hit enter and followed by string to get reverse of string)
- *factor 111* (self-explanatory)
- *df* (disk free command, it's a file system command that tells disk usage)
  - -h (human readable), -k (in KBs), -m (in MBs)
- *du* (disk usage of file or folder)
  - -h (human readable), -k (KB), -m (MB)
- *ncdu* (Interactive disk usage analyzer)
- *diff file1.txt file2.txt* (compares and shows diff b/w 2 files)
- *alias ipconfig="ifconfig"* (self-explanatory doskey ip=ipconfig for windows)
  - *unalias ipconfig* (to unlias it)
  - The alias will be removed after we exit or logout, to permanently keep it save it in bashrc hidden file **(/home/user/.bashrc**), it's a special file that is run at very beginning when system starts so we can add our aliases and variables at bottom here, then restart the system so that .bashrc is executed or just simply use '*. ~/.bashrc*' it's actually ./home/kartikey/.bashrc (i.e. a way to execute)
- *apropos copy* (Tells all commands related to word 'copy')
- *uptime* (tells uptime)
  - -p (prettify), -s (since)
- *last* (tells every system login and reboots)
- *less /etc/passwd* (lists users but not all)

- *tail /var/log/auth.log* (opens last 10 lines of auth.log, similarly can use head)
- *sudo tail -f* ***/var/log/auth.log*** (real time constant logging for debugging)
- *history* (To see history of commands like doskey /history in cmd)
    - -c (clears history)
- *lsblk* (list blocks i.e., HDDs)
- *lsusb* (list USBs)
- *lsof* (list open files)
- *twitter="Elon Musk"* the twitter is not a variable but it's temporary and limited to terminal only, so to make is globally accessible by child processes i.e scripts we create use *export twitter*. If we want this to be permanent add *export twitter="Elon Musk"* @ end of .bashrc file
- *shutdown now* (obvious)
- *open ./coolfile.jpg* (opens the file with default prog for that extension)
- *reboot*
- *google-chrome* *www.fb.com* (opens website or simply use google-chrome)
- *yes demo_text* (Spams demo_text interminal)
- *nautilus* (opens file explorer)
- *code .* (opens vscode in the current directory)
- *subl .* (opens sublime in the current directory)
- *xdg-open file.txt* (opens file in GUI text editor or use *gedit file.txt*)
- *eject* (Ejects the disk)

10. Funny and cool commands:

- *oneko* (Cat follows mouse cursor)
- *cmatrix* (matrix screensaver)
- *sl* (steam locomotive)
- *nyancat* (u guessed it)
- *fortune | cowsay -f tux* (combines fortune and cowsay but ascii is tux, see cowsay -l to get list of asciis)
- *toilet -f mono12 -F metal/gay kartikey* (shows stylish font)
- Curl Commands:

    - *curl parrot.live* (shows dancing parrot, needs curl to be installed, since curl is also on windows it'll also work there)
    - *curl ascii.live/forest*
- *lolcat* (makes text colorful)
    - *echo "hey my niggas, wassup" | lolcat*
    - *fortune | lolcat*
    - *fortune | cowsay | lolcat*
- *bb* (ASCII movie)
- *telnet towel.blinkenlights.nl* (ASCII Star Wars)
- *xeyes* (eye window that follows cursor)
- *xcowsay* (graphical cowsay)
- *figlet "hello nigs"* (Cool graphics text, but toilet is better)

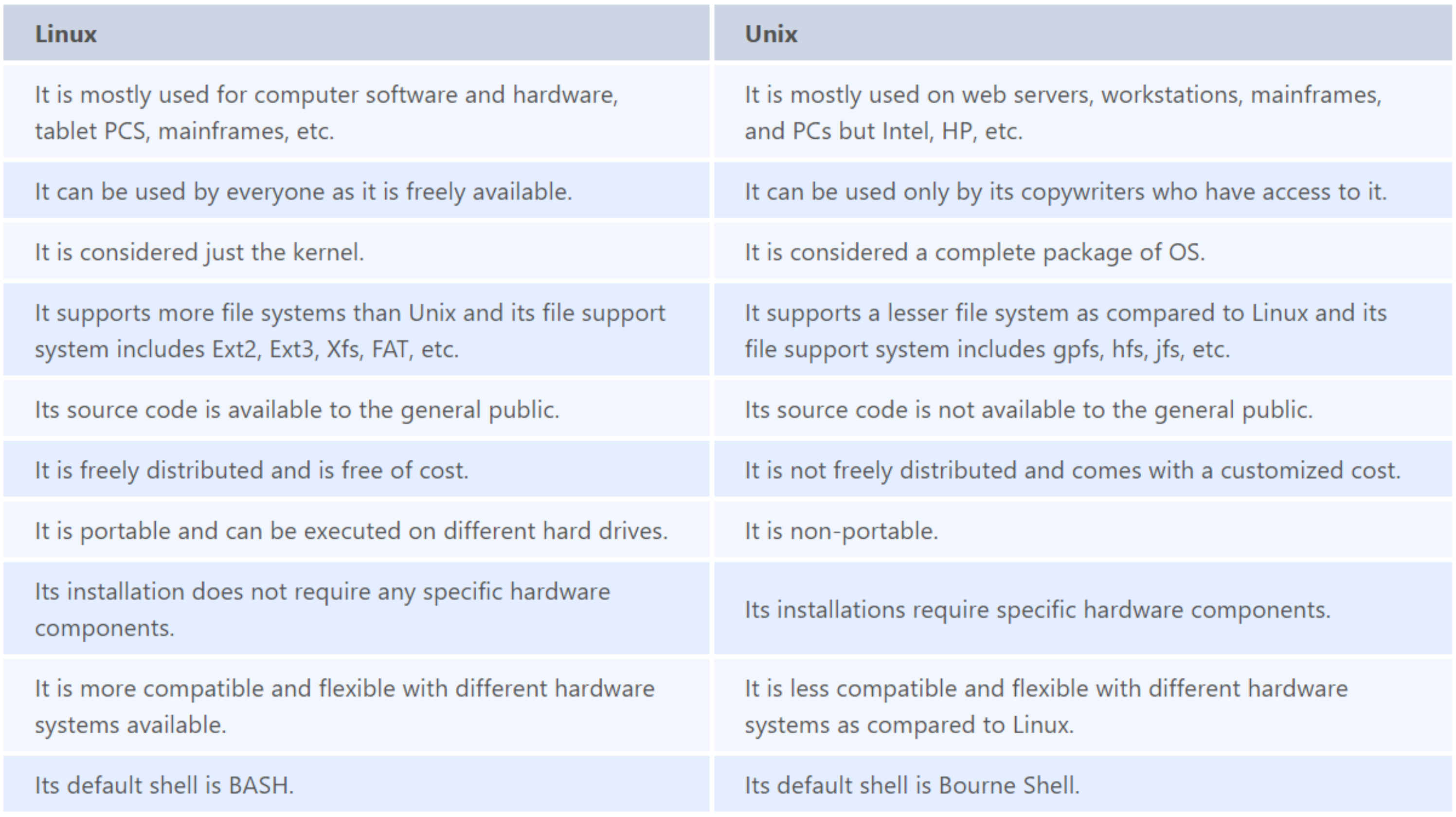
11. Basic Bash Scripting:

- One Liner: *while true; do eject; sleep 1; done &*

```
#!bin/bash   #this is a must in bash file to make sure which shell we'll use
echo "Enter name: "
read x
age=$1 #age value will be extracted from 1st argument
cgpa=$2 #cg value will be extracted from 2nd argument
user=$(whoami) #syntax for variable=o/p from a command
os=$(uname)
y="Kartikey"
echo "Wassup $x! $y is your BOSS!"
sleep 2
echo "Hang tight my nigga $x!"
sleep 1
echo "Begin $age years old and scoring $cgpa CGPA is great!"
```

```
sleep 2

echo "Looks like $user likes to use $os"

sleep 1

echo "Your current working directory is $PWD and hostname is
$HOSTNAME"

sleep 2

echo "Calculating your rich age:..."

sleep 1

rich=$(( ($RANDOM%15) + $age))

echo "You'll be millionare by the age of $rich"

sleep 2

echo "Are you satisfied?(y/n)"

read status

if [[ $status == "y" ]]; then

    echo "Cool, bbye"

    exit 1 #exit with exit code 1

else

    echo "You'll be millionare by the age of $age"

fi
```

12. Linux vs Unix:



Linux	Unix
It is mostly used for computer software and hardware, tablet PCS, mainframes, etc.	It is mostly used on web servers, workstations, mainframes, and PCs but Intel, HP, etc.
It can be used by everyone as it is freely available.	It can be used only by its copywriters who have access to it.
It is considered just the kernel.	It is considered a complete package of OS.
It supports more file systems than Unix and its file support system includes Ext2, Ext3, Xfs, FAT, etc.	It supports a lesser file system as compared to Linux and its file support system includes gpfs, hfs, jfs, etc.
Its source code is available to the general public.	Its source code is not available to the general public.
It is freely distributed and is free of cost.	It is not freely distributed and comes with a customized cost.
It is portable and can be executed on different hard drives.	It is non-portable.
Its installation does not require any specific hardware components.	Its installations require specific hardware components.
It is more compatible and flexible with different hardware systems available.	It is less compatible and flexible with different hardware systems as compared to Linux.
Its default shell is BASH.	Its default shell is Bourne Shell.