WEEK-2

Topics: Virtualization technology, working, types, Potentials and challenges of Virtualization, Virtual Machines, Containers. Linux Boot process. Linux command line - Interpreter, shell, CLI over GUI, Types of user's super and normal, Linux user manual.

Virtualization technology

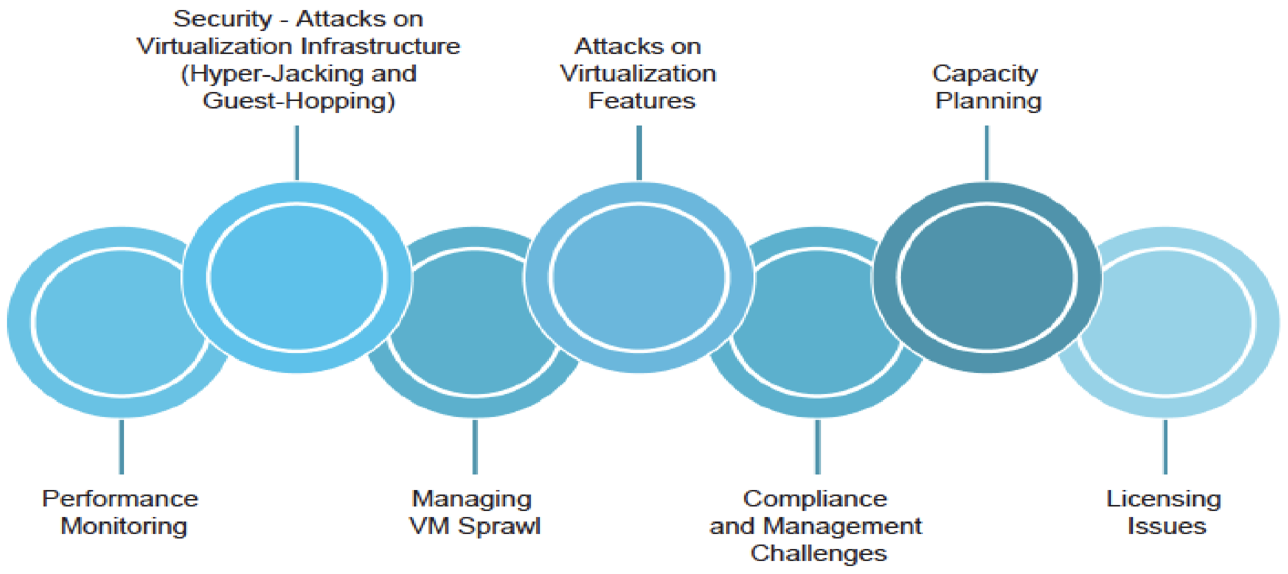
Virtualization is technology that allows you to create multiple simulated environments or dedicated resources from a single, physical hardware system. Software called a hypervisor connects directly to that hardware and allows you to split 1 system into separate, distinct, and secure environments known as virtual machines (VMs). These VMs rely on the hypervisor's ability to separate the machine's resources from the hardware and distribute them appropriately.

The physical hardware, equipped with a hypervisor, is called the host, while the many VMs that use its resources are guests. These guests treat computing resources—like CPU, memory, and storage—as a pool of resources that can easily be relocated.

Benefits of Virtualization

- **Partitioning:** Multiple applications and operating systems can be supported within a single physical system Servers can be consolidated into virtual machines on either a scale-up or scale-out architecture Computing resources are treated as a uniform pool to be allocated to virtual machines in a controlled manner
- **Isolation:** Virtual machines are completely isolated from the host machine and other virtual machines. If a virtual machine crashes, all others are unaffected Data does not leak across virtual machines and applications can only communicate over configured network connections.
- **Encapsulation:** Complete virtual machine environment is saved as a single file; easy to back up, move and copy Standardized virtualized hardware is presented to the application - guaranteeing compatibility.

The Challenges Of Virtualization


Security - Attacks on Virtualization Infrastructure (Hyper-Jacking and Guest-Hopping)
Attacks on Virtualization Features
Capacity Planning
Performance Monitoring
Managing VM Sprawl
Compliance and Management Challenges
Licensing Issues

Performance Monitoring: Unlike physical servers, monitoring the performance of the virtual servers requires a different approach. Conventional CPU and memory utilization monitoring don't work. In a virtual infrastructure, the VMs share the common hardware resources such as CPU, memory, and storage. Metrics such as CPU ready, memory ready, memory balloon, and swapped memory have to be monitored across all the VMs.

Security: In virtualized systems, the hypervisor is the single point of failure in security and if you lose it, you lose the protection of sensitive information. So the best way to avoid hyper-jacking is to use hardware-rooted trust and secure launching of the hypervisor.

Managing VM Sprawl: The virtual infrastructure has to be monitored continuously to identify VMs that are idle for a certain period, VMs that have resources overprovisioned or under-provisioned, and VMs that are not even powered on since they were provisioned. By removing such unused VMs – or optimizing the resources allocated to VMs – virtual sprawl can be greatly minimized.

Attacks on Virtualization features: Although there are multiple features of virtualization that can be targeted for exploitation, the more common targets include VM migration and virtual networking functions. VM migration, if done insecurely, can expose all aspects of a given VM to both passive sniffing and active manipulation attacks.

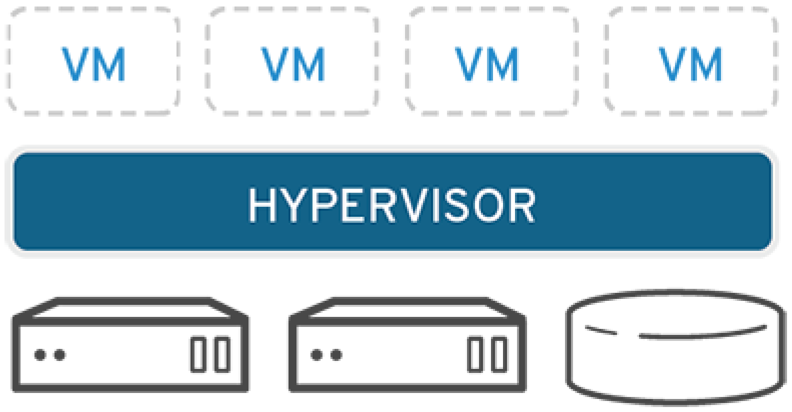
Compliance and Management Challenges: Compliance auditing and enforcement, as well as day-to-day system management, are challenging issues when dealing with virtualized systems. VM sprawl will make it a challenge to get accurate results from vulnerability assessments.

Capacity Planning: Without proper capacity planning, the server management team will have a hard time scaling the infrastructure to meet business demands. Capacity planning also requires constant monitoring of the virtual infrastructure. Plan how many resources have to be added to handle the future load.

Licensing Issues: Software vendors license virtualization in different ways. For example, one organization was running seven virtual instances on a server with four physical cores. Depending on which software they were using, they required seven licenses from one vendor – but only four from another.

How does virtualization works:

Software called hypervisors separate the physical resources from the virtual environments—the things that need those resources. Hypervisors can sit on top of an operating system (like on a laptop) or be installed directly onto hardware (like a server), which is how most enterprises virtualize. Hypervisors take your physical resources and divide them up so that virtual environments can use them.

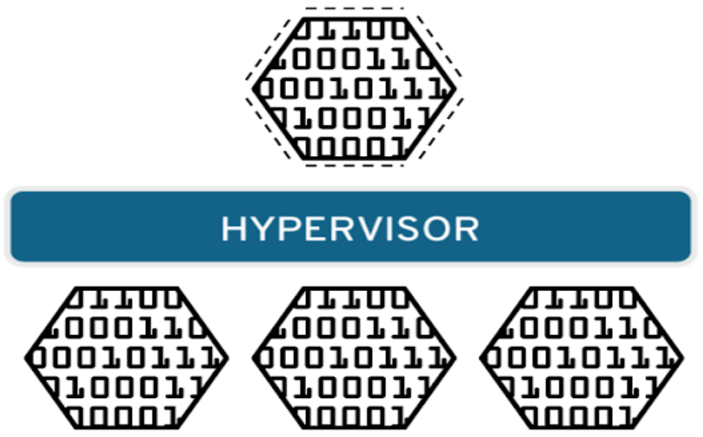


Resources are partitioned as needed from the physical environment to the many virtual environments. Users interact with and run computations within the virtual environment (typically called a guest machine or virtual machine). The virtual machine functions as a single data file. And like any digital file, it can be moved from one computer to another, opened in either one, and be expected to work the same.

When the virtual environment is running and a user or program issues an instruction that requires additional resources from the physical environment, the hypervisor relays the request to the physical system and caches the changes—which all happens at close to native speed (particularly if the request is sent through an open source hypervisor based on KVM, the Kernel-based Virtual Machine)

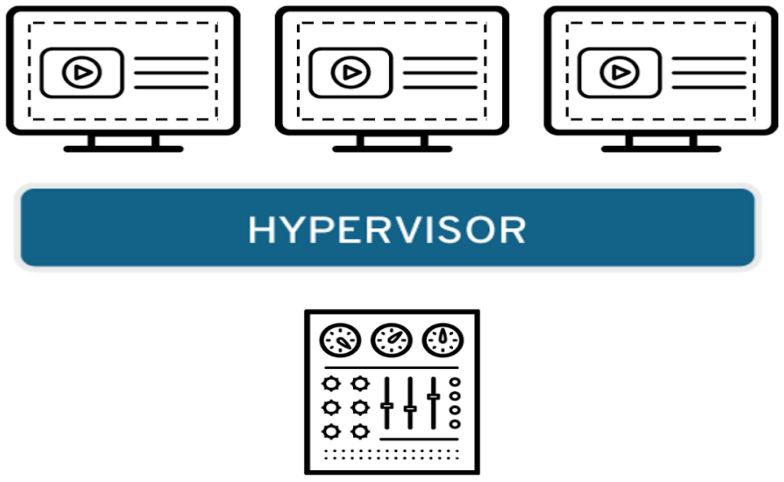
Types of Virtualization

Data Virtualization



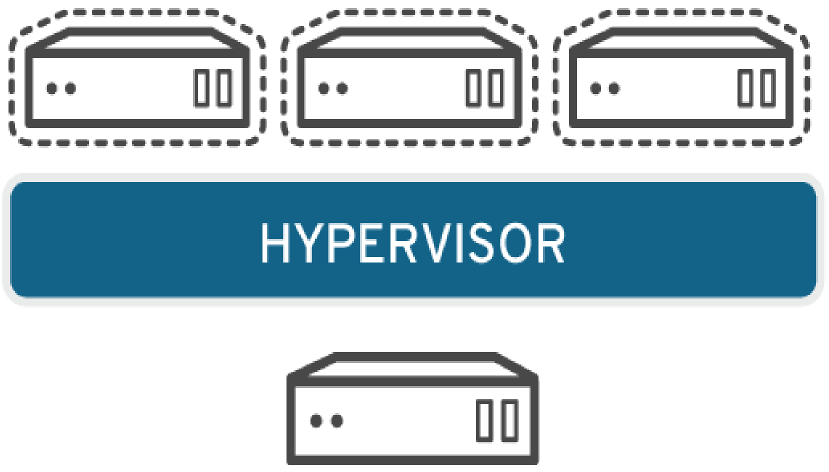
Data that's spread all over can be consolidated into a single source. Data virtualization allows companies to treat data as a dynamic supply—providing processing capabilities that can bring together data from multiple sources, easily accommodate new data sources, and transform data according to user needs. Data virtualization tools sit in front of multiple data sources and allows them to be treated as single source, delivering the needed data—in the required form—at the right time to any application or user.

Desktop virtualization



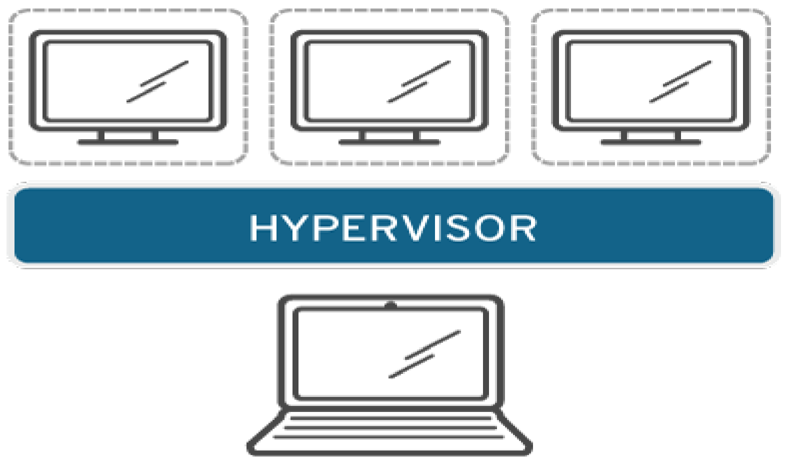
Easily confused with operating system virtualization—which allows you to deploy multiple operating systems on a single machine—desktop virtualization allows a central administrator (or automated administration tool) to deploy simulated desktop environments to hundreds of physical machines at once. Unlike traditional desktop environments that are physically installed, configured, and updated on each machine, desktop virtualization allows admins to perform mass configurations, updates, and security checks on all virtual desktops.

Server virtualization



Servers are computers designed to process a high volume of specific tasks really well so other computers—like laptops and desktops—can do a variety of other tasks. Virtualizing a server lets it to do more of those specific functions and involves partitioning it so that the components can be used to serve multiple functions.

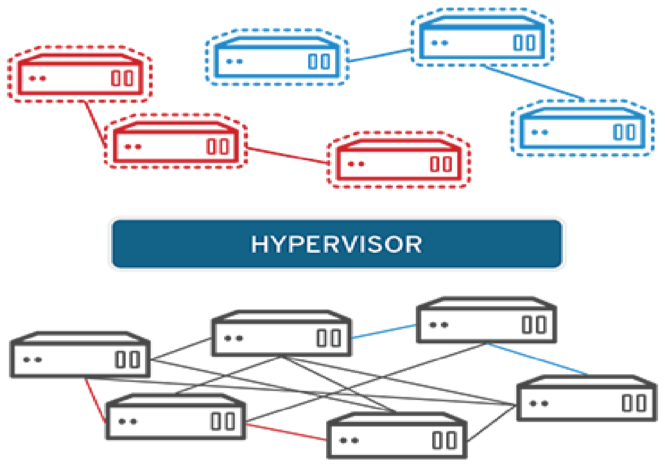
Operating system virtualization



Operating system virtualization happens at the kernel—the central task managers of operating systems. It's a useful way to run Linux and Windows environments side-by-side. Enterprises can also push virtual operating systems to computers, which:

- Reduces bulk hardware costs, since the computers don't require such high out-of-the-box capabilities.
- Increases security, since all virtual instances can be monitored and isolated.
- Limits time spent on IT services like software updates.

Network functions virtualization



Network functions virtualization (NFV) separates a network's key functions (like directory services, file sharing, and IP configuration) so they can be distributed among environments. Once software functions are independent of the physical machines they once lived on, specific functions can be packaged together into a new network and assigned to an environment. Virtualizing networks reduces the number of physical components—like switches, routers, servers, cables, and hubs—that are needed to create multiple, independent networks, and it's particularly popular in the telecommunications industry.

Virtual Machines

A Virtual Machine is the representation of a physical machine by software. It has its own set of virtual hardware (e.g., RAM, CPU, NIC, hard disks, etc.) upon which an operating system and applications are loaded. The operating system sees a consistent, normalized set of hardware regardless of the actual physical hardware components. Software called a hypervisor separates the machine's resources from the hardware and provisions them appropriately so they can be used by the VM

Virtual Machines are not emulators are simulators. They are real machines that can do the same things physical computers can do and more. Because of the flexibility of virtual machines, physical machines become less a way to provide services (application, databases etc.) and more a way to house the virtual machines that provide those resources. Servers are easy to manage if they are on virtual machines. Virtual machines also reduce cost of hardware, maintenance and environment support.

Containers

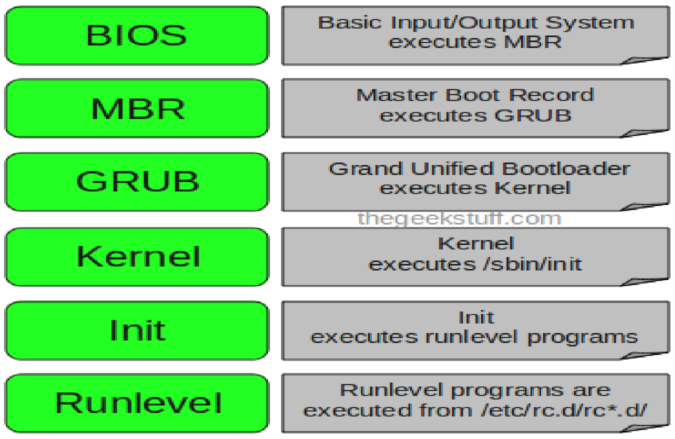
Linux containers and virtual machines (VMs) are packaged computing environments that combine various IT components and isolate them from the rest of the system. Their main differences are in terms of scale and portability.

Containers are typically measured by the megabyte. They don't package anything bigger than an app and all the files necessary to run, and are often used to package single functions that perform specific tasks (known as a microservice). The lightweight nature of containers—and their shared operating system (OS)—makes them very easy to move across multiple environments.

VMs are typically measured by the gigabyte. They usually contain their own OS, allowing them to perform multiple resource-intensive functions at once. The increased resources available to VMs allow them to abstract, split, duplicate, and emulate entire servers, OSs, desktops, databases, and networks.

Linux boot process

The following are the 6 high level stages of a typical Linux boot process.


BIOS
Basic Input/Output System executes MBR
MBR
Master Boot Record executes GRUB
GRUB
Grand Unified Bootloader executes Kernel
thegeekstuff.com
Kernel
Kernel executes /sbin/init
Init
Init executes runlevel programs
Runlevel
Runlevel programs are executed from /etc/rc.d/rc*.d/

1. BIOS

- BIOS stands for Basic Input/Output System
- Performs some system integrity checks
- BIOS loads and executes the MBR boot loader.

2. MBR

- MBR stands for Master Boot Record.
- It is located in the 1st sector of the bootable disk. Typically /dev/hda, or /dev/sda
- MBR is less than 512 bytes in size. This has three components
- 1) primary boot loader info in 1st 446 bytes
- 2) partition table info in next 64 bytes
- 3) MBR validation check in last 2 bytes.
- It contains information about GRUB (or LILO in old systems).
- So, in simple terms MBR loads and executes the GRUB boot loader.

3. GRUB

- GRUB stands for Grand Unified Bootloader.
- If you have multiple kernel images installed on your system, you can choose which one to be executed.
- GRUB displays a splash screen, waits for few seconds, if you don't enter anything, it loads the default kernel image as specified in the grub configuration file.
- GRUB has the knowledge of the filesystem (the older Linux loader LILO didn't understand filesystem).
- Grub configuration file is /boot/grub/grub.conf (/etc/grub.conf is a link to this). The following is sample grub.conf of CentOS.

4. Kernel

- Mounts the root file system as specified in the "root=" in grub.conf
- Kernel executes the /sbin/init program

- Since init was the 1st program to be executed by Linux Kernel, it has the process id (PID) of 1. Do a ‘ps -ef | grep init’ and check the pid.
- initrd stands for Initial RAM Disk.
- initrd is used by kernel as temporary root file system until kernel is booted and the real root file system is mounted. It also contains necessary drivers compiled inside, which helps it to access the hard drive partitions, and other hardware.

5. Init

- Looks at the /etc/inittab file to decide the Linux run level.
- Following are the available run levels
- 0 – halt
- 1 – Single user mode
- 2 – Multiuser, without NFS
- 3 – Full multiuser mode
- 4 – unused
- 5 – X11
- 6 – reboot
- Init identifies the default initlevel from /etc/inittab and uses that to load all appropriate program.
- Execute ‘grep initdefault /etc/inittab’ on your system to identify the default run level
- If you want to get into trouble, you can set the default run level to 0 or 6. Since you know what 0 and 6 means, probably you might not do that.
- Typically, you would set the default run level to either 3 or 5.

6. Runlevel programs

- When the Linux system is booting up, you might see various services getting started. For example, it might say “starting sendmail …. OK”. Those are the runlevel programs, executed from the run level directory as defined by your run level.
- Depending on your default init level setting, the system will execute the programs from one of the run level (0 to 6) directories (/etc/rc.d/rc{0-6}.d/)
- Under the /etc/rc.d/rc*.d/ directories, you would see programs that start with S and K.

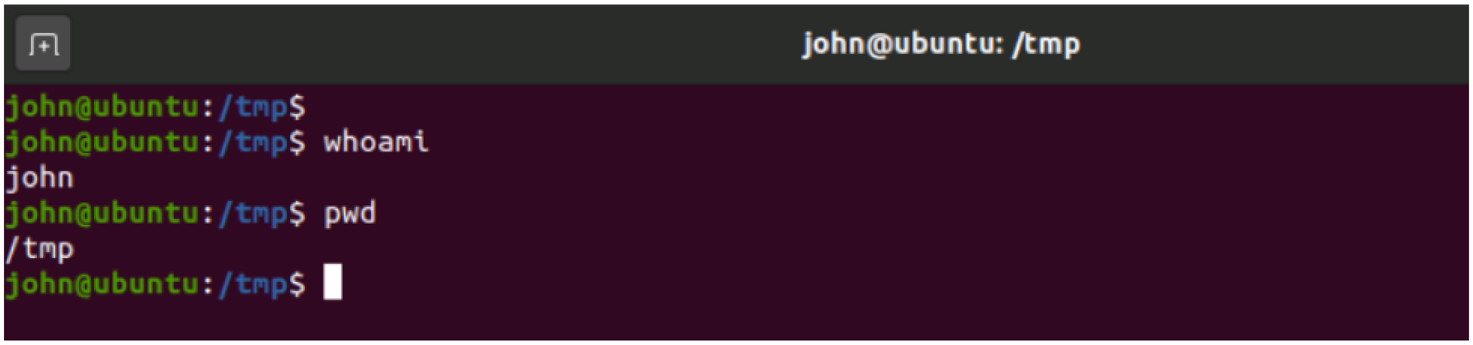
- Programs starts with S are used during startup. S for startup.
- Programs starts with K are used during shutdown. K for kill.

That is what happens during the Linux boot process.

Linux command line

The Linux command line is a text interface to your computer. Also known as shell, terminal, console, command prompts and many others, is a computer program intended to interpret commands.
Allows users to execute commands by manually typing at the terminal, or has the ability to automatically execute commands which were programmed in “Shell Scripts”.

First look at the command line



```
john@ubuntu:/tmp$
john@ubuntu:/tmp$ whoami
john
john@ubuntu:/tmp$ pwd
/tmp
john@ubuntu:/tmp$
```

When a terminal is open, it presents you with a prompt.
Let's analyze the screenshot above:
Line 1: The shell prompt, it is composed by *username@hostname:location$*

Username: our username is called “john”

Hostname: The name of the system we are logged on

Location: the working directory we are in

$: Delimits the end of prompt

After the $ sign, we can type a command and press Enter for this command to be executed.
Line 2: After the prompt, we have typed the command *whoami* which stands for “who am i“ and pressed [Enter] on the keyboard.
Line 3: Shows us the result of the *whoami* command we have previously issued, also known as command output. This command simply prints out the username of the current user.
Line 4: Shows an example of another basic command called *pwd* which stands for print working directory.

Line 5: As seen before, this line shows the result of the command previously issued. /tmp is our working directory.

Line 6: Presents us with a new prompt, and waits for us to type a new command.
There are a few important things to keep in mind when using a Linux shell:

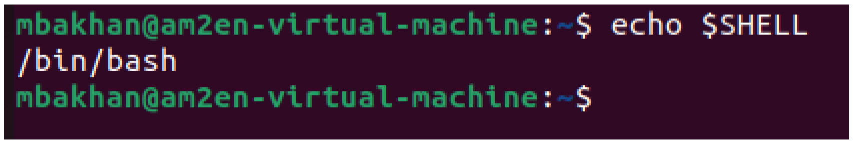
- It is case sensitive
- The / (forward-slash) is a special character used as directory separator
- Nearly every Linux command supports --help argument
- File extensions don’t matter

Common Types of Shells in Linux

1. The Bourne **Sh**ell (sh - # and $) was originally developed in 1979 by Stephen Bourne while working at Bell Labs.
2. The **GNU B**ourne **A**gain **Sh**ell (bash – bash-versionnumber# and bash-versionnumber$) was written as a free and open source replacement for the Bourne Shell.
3. The **C Shell (csh –** hostname# and hostname$**)** was created at university of California by Bill Joy to include useful programming features.
4. The **Korn Shell (ksh -** # and $) was developed at AT & T Bell Labs by David Korn, to improve the Bourne Shell.

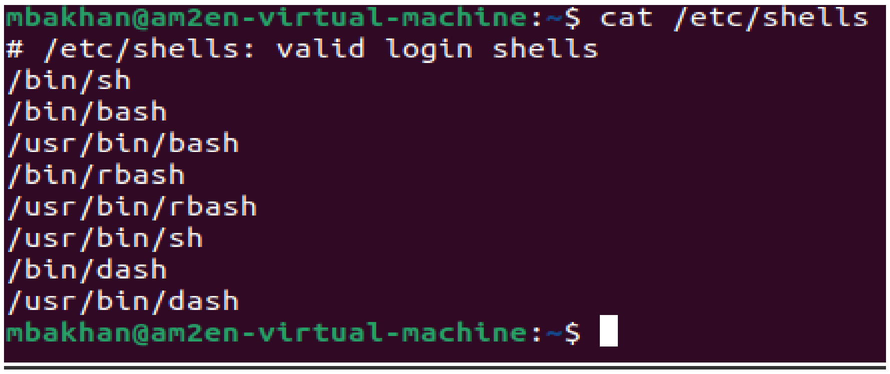
All the above shells are stored in Linux Directories - /bin/sh, /bin/bash, /bin/csh and /bin/ksh respectively.

To check your present login shell, use the below command



```
mbakhan@am2en-virtual-machine:~$ echo $SHELL
/bin/bash
mbakhan@am2en-virtual-machine:~$
```

To check the available shells in the system



```
mbakhan@am2en-virtual-machine:~$ cat /etc/shells
# /etc/shells: valid login shells
/bin/sh
/bin/bash
/usr/bin/bash
/bin/rbash
/usr/bin/rbash
/usr/bin/sh
/bin/dash
/usr/bin/dash
mbakhan@am2en-virtual-machine:~$
```

Linux user types

- There are two types of users – the **root or super user** and **normal users**.
- A root or super user can access all the files, while the normal user has limited access to files. In computing, the superuser is a special user account used for system administration. Depending on the operating system (OS), the actual name of this account might be root, administrator, admin or supervisor.
- Normal users are the users created by the root or another user with sudo privileges. Usually, a normal user has a real login shell and a home directory. Each user has a numeric user ID called UID.

A super user can add, delete and modify a user account. The full account information is stored in the */etc/passwd* file and a hash password is stored in the file */etc/shadow*.

Interpreters: A command line interpreter is any program that allows the entering of commands and then executes those commands to the operating system. It's literally an interpreter of commands.

CLI over GUI

CLI : Command line Interface, allows the users to interact with the system using commands.

GUI : Graphical User Interface, allows the users to interact with the system using graphical elements such as windows, icons, menus

Sl no	CLI	GUI
1	CLI is difficult to use.	Whereas it is easy to use
2	It consumes low memory.	While consumes more memory
3	In CLI we can obtain high precision.	While in it, low precision is obtained
4	CLI is faster than GUI	The speed of GUI is slower than CLI.
5	CLI's appearance can not be modified or changed.	While it's appearance can be modified or changed.
6	CLI operating system needs only keyboard.	While GUI operating system need both mouse and keyboard.
7	In CLI, input is entered only at command prompt.	While in GUI, input can be entered anywhere on the screen.
8	In CLI, the information is shown or presented to the user in plain text and files	While in GUI, the information is shown or presented to the user in any form such as: plain text, videos, images, etc.
9	In CLI, there are no menus provided.	While in GUI, menus are provided.
10	There are no graphics in CLI.	While in GUI, graphics are used.
11	CLI do not use any pointing devices.	While it uses pointing devices for selecting and choosing items.
12	In CLI, spelling mistakes and typing errors are not avoided	Whereas in GUI, spelling mistakes and typing errors are avoided.

Linux User Manual

man command in Linux is used to display the user manual of any command that we can run on the terminal. It provides a detailed view of the command which includes NAME, SYNOPSIS, DESCRIPTION, OPTIONS, EXIT STATUS, RETURN VALUES, ERRORS, FILES, VERSIONS, EXAMPLES, AUTHORS and SEE ALSO.

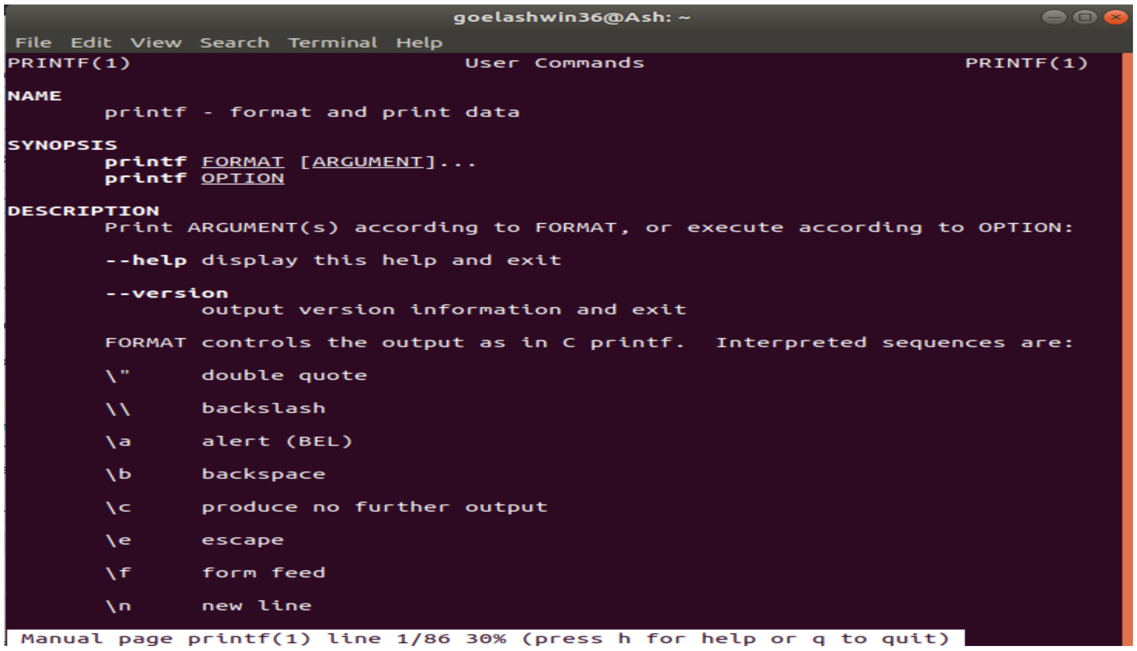
Every manual is divided into the following sections:

- Executable programs or shell commands
- System calls (functions provided by the kernel)
- Library calls (functions within program libraries
- Games
- Special files (usually found in /dev)
- File formats and conventions eg /etc/passwd
- Miscellaneous (including macro packages and conventions), e.g. groff(7)
- System administration commands (usually only for root)
- Kernel routines [Non standard]

Syntax :

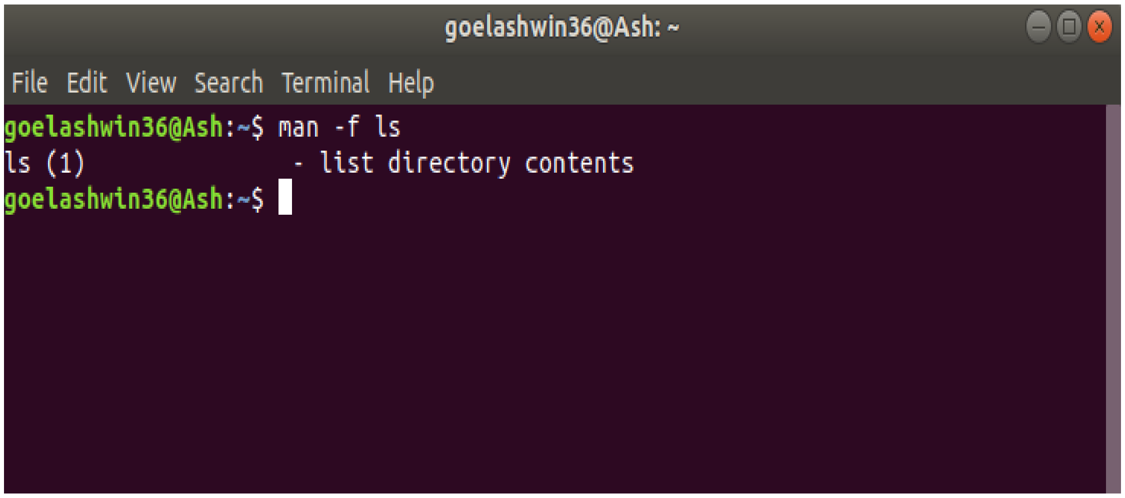
$man [OPTION]... [COMMAND NAME]...

Example: 1. $ man printf



```
PRINTF(1)                        User Commands                        PRINTF(1)

NAME
       printf - format and print data

SYNOPSIS
       printf FORMAT [ARGUMENT]...
       printf OPTION

DESCRIPTION
       Print ARGUMENT(s) according to FORMAT, or execute according to OPTION:

       --help display this help and exit

       --version
              output version information and exit

       FORMAT controls the output as in C printf.  Interpreted sequences are:

       \"     double quote

       \\     backslash

       \a     alert (BEL)

       \b     backspace

       \c     produce no further output

       \e     escape

       \f     form feed

       \n     new line
 Manual page printf(1) line 1/86 30% (press h for help or q to quit)
```

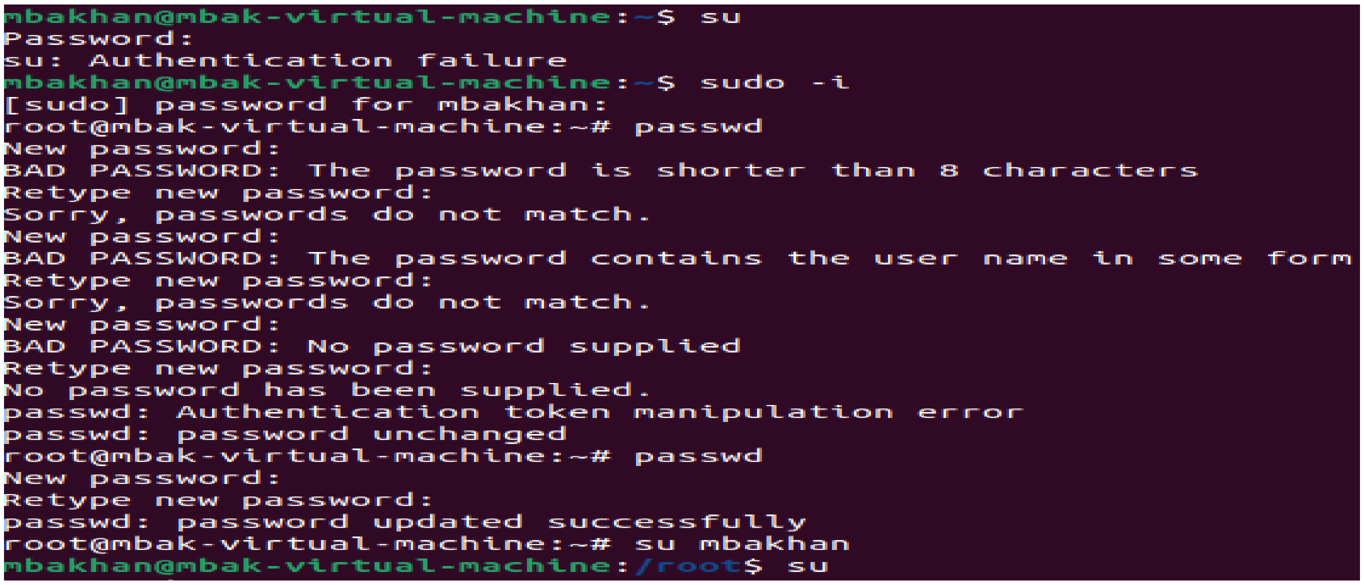
2. man -f ls



```
goelashwin36@Ash:~$ man -f ls
ls (1)               - list directory contents
goelashwin36@Ash:~$ 
```

Few Commands to Try!

1. **cat /etc/shells --- **To Know the Valid Login Shells available for the system.
2. **whoami ---** To know the currently logged in user.
3. **pwd ---** To know the Present Working Directory.
4. **su ---** To switch user or substitute {This command needs root password which is locked in Ubuntu}.
5. **sudo** –i **---** Alternative to su command to get privileges of root (administrator).
6. **passwd ---** To set or change the password of root.
7. **su username or exit or ctrl+d or logout ---** To return to user account.



```
mbakhan@mbak-virtual-machine:~$ su
Password:
su: Authentication failure
mbakhan@mbak-virtual-machine:~$ sudo -i
[sudo] password for mbakhan:
root@mbak-virtual-machine:~# passwd
New password:
BAD PASSWORD: The password is shorter than 8 characters
Retype new password:
Sorry, passwords do not match.
New password:
BAD PASSWORD: The password contains the user name in some form
Retype new password:
Sorry, passwords do not match.
New password:
BAD PASSWORD: No password supplied
Retype new password:
No password has been supplied.
passwd: Authentication token manipulation error
passwd: password unchanged
root@mbak-virtual-machine:~# passwd
New password:
Retype new password:
passwd: password updated successfully
root@mbak-virtual-machine:~# su mbakhan
mbakhan@mbak-virtual-machine:/root$ su
```