Chapter 3 :- Linux Basic and System Startup

* Learning objectives

→ By the end of this chapter, you should be able to:

- Identify Linux file systems.
- Identify the differences between Partitions and filesystems.
- Describe the boot Process.
- Install Linux on a computer.

* The Boot process

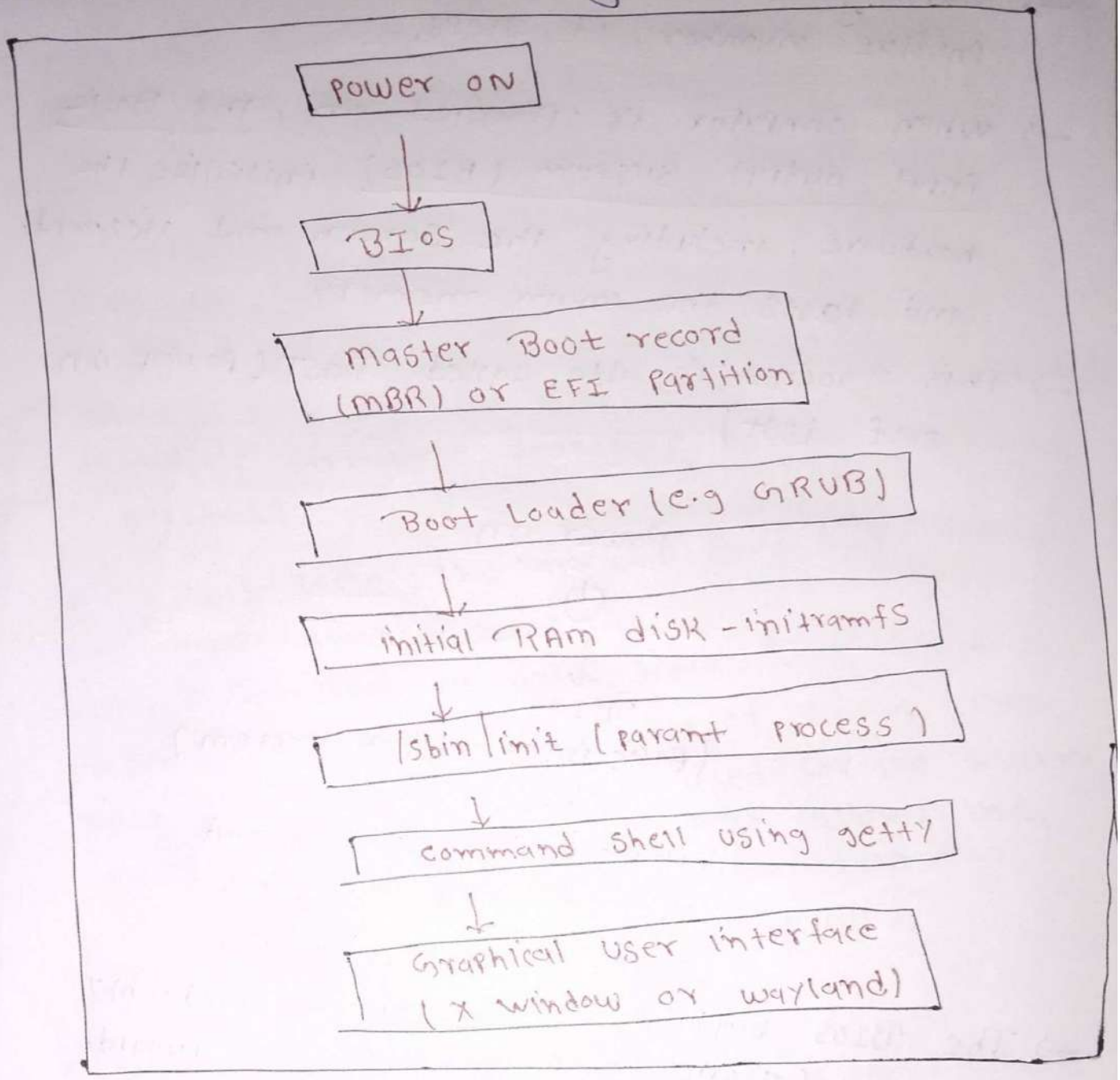
→ The Linux boot process is procedure for initializing the system.

→ It consists of everything that happens from when the computer Power is first switched on until the user interface is fully operational.

→ Having a good understanding of the steps in the boot process may help you with troubleshooting Problem, as well as with tailoring the computer's Performance to your needs.

→ on the other hand, the bout Process can be rather technical and you can start

using Linux without knowing all the details



The Boot Process

* BIOS - The First step

-> while Linux runs on many kind of hardware, we will concentrate on the X86 family, which is basic of almost all the desktop and laptop PCs.

→ Starting an X86 - based Linux system involve number of steps.

→ when computer is Powered on, the Basic input output System (BIOS) initialize the hardware, including the screen and keyword, and tests the main memory

→ This Process is also called POST (Power on self test)

Power on

↓

BIOS

(Basic input output system)

↓

initializes the screen and keyword and test the main memory BIOS

→ The BIOS Software is stored on read-only memory (ROM) chip on the motherboard.

→ After this, the remainder of boot Process is controlled by operating System (OS).

* master Boot Record (MBR) or EFI Partitions

-> once the POST is completed, system control passes from the BIOS to the boot loader.

-> The Boot loader is usually stored on one of the system storage devices, such as hard disk or SSD drive, Either in the boot sector (for traditional BIOS/MBR System) or the EFI partition (for most recent (unified) Extensible Firmware interface or EFI/UEFI systems).

-> UP to this stage, the machine does not access any mass storage media.

-> Then, information on date, time, and most important peripherals are loaded from CMOS value (After a technology used for the battery - powered memory stored, which allows the system to keep track of the date and time even when it is powered off).

-> A number of boot loader exist for Linux; the most common ones are GRUB (for GRand Unified Boot loader

-> ISOLINUX (for booting from removable media)

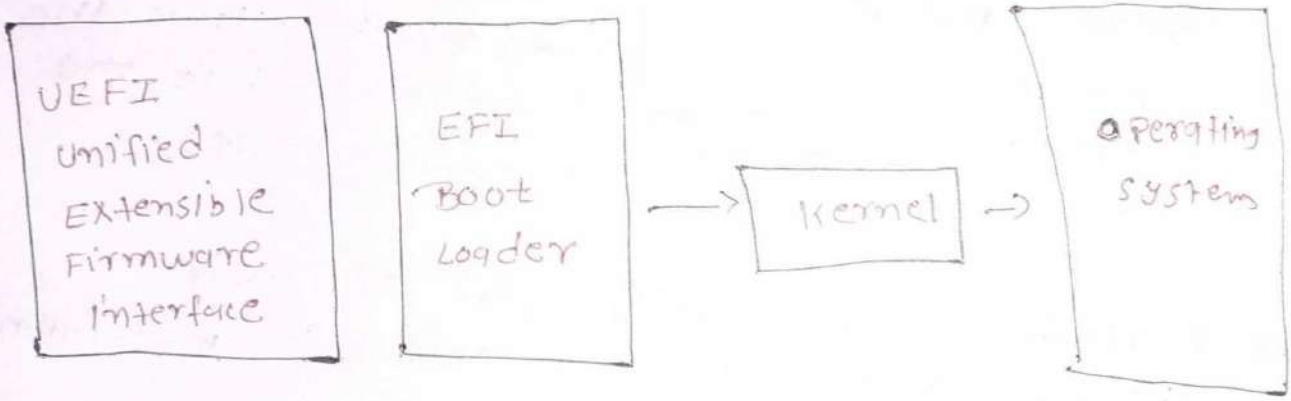
-> and DAS U-Boot (for Booting on embedded device application)

-> most Linux boot loaders can present a user interface for choosing alternative option for booting Linux and even other operating system that might be installed.

→ when booting Linux, the boot loader is responsible for loading kernel image and the initial RAM disk or file system (which contains some critical files and device drivers to start the system) into memory.

* BIOS Boot

* UEFI Boot



* Boot Loader in Action

→ The boot loader has two distinct stages;

→ For System using BIOS | MBR method, the boot loader resides at the first sector of the hard disk, also known as master Boot Record (MBR)

→ The size of MBR is just 512 bytes.

→ In this stage the boot loader examines the Partition table and find a bootable Partition.

→ once it finds bootable Partition, it then searches for second stage boot loader, for Example GRUB and loads it into RAM.

→ For systems using the EFI | UEFI method, UEFI firmware reads it's Boot manager data to determine which UEFI application is to be launched and from where (from disk and Partition the EFI partition can be found)

→ The firmware then launches the UEFI application for example GRUB as defined in boot entry in firmware's boot manager. This Procedure is more complicated but more versatile than older MBR methods.

→ The second stage boot loader resides under /boot.

→ A splash screen is displayed, which allows us to choose which operating system and/or kernel to boot.

→ After the OS and kernel are selected, the boot Loader loads the kernel of operating system into RAM and passes control to it.

→ Kernels are almost always compressed, so the first job they have is to uncompress themself

→ After this, it will check and analyze the system hardware and initialize any hardware device drivers built into the kernel.

* The initial RAM Disk

-> The initramfs filesystem image contains Program and binary files that Perform all action needed to mount the Proper root file system, including Providing the kernel functionality required for specific file system that will be used, and loading the device drivers for mass storage controllers, by taking advantage of udev system (for user device) -> which is responsible for figuring out which device are present, locating the device driver they need to operate properly, and loading them.

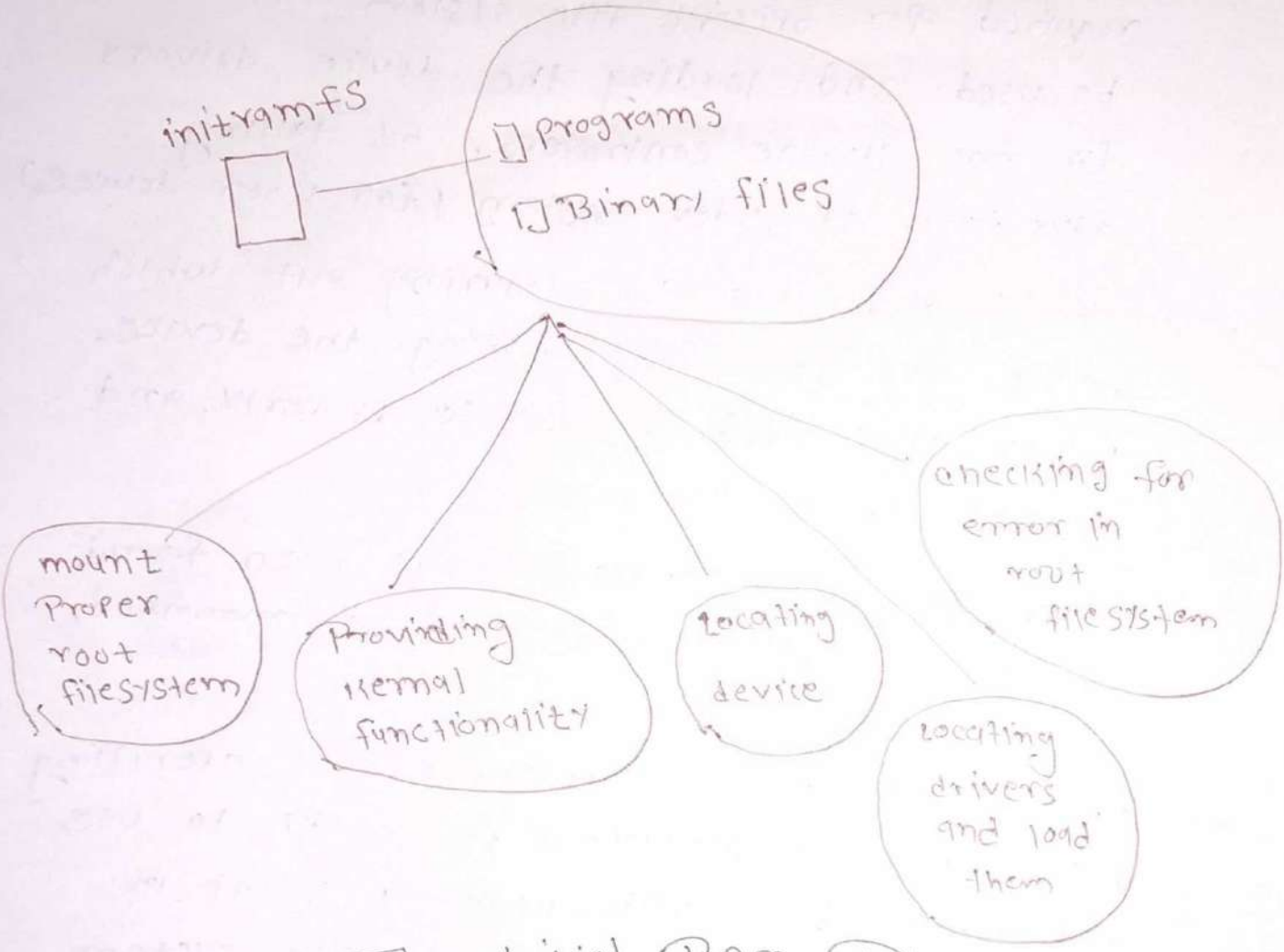
-> After the root filesystem has been found, It is checked for error and, mounted.

-> The mount Program instruct the operating system that a filesystem is ready to use and associated it with particular Point in the overall hierachy of the file system (the mount Point).

-> If this is successful, the initramfs is cleared from RAM, and the init Program on the root filesystem (/sbin/init) is executed.

→ init handles the mounting and Pivoting over to the final real root file system.

→ If special hardware drivers are needed ~~the~~ before the mass storage can be accessed, they must be in the initramfs image.



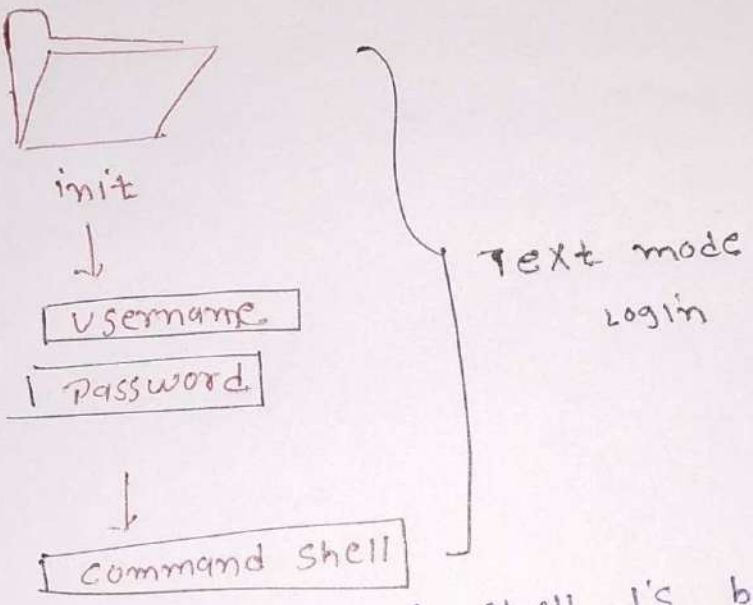
The initial RAM Disk.

* Text - mode Login

→ Near the end of boot process, init starts a number of text-mode login prompts.

→ These enable you to type your username, followed by your password and eventually get a command shell.

→ However, if you are running a system with graphical login interface, you will not see these at first.



→ usually, the default command shell is bash but there are number of other advanced command shells available.

→ The shell prints a text prompt, indicating it is ready to accept commands;

→ After the user type the command and press Enter, the command is executed and another prompt is displayed after the command is done.